

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ІНТЕЛЕКТУАЛЬНОГО САЙТУ-ЧАТУ ДЛЯ СТУДЕНТІВ: ПОКРАЩЕННЯ
НАВЧАЛЬНОГО ПРОЦЕСУ. RESEARCH AND SOFTWARE IMPLEMENTATION
OF A SMART CHAT SITE FOR STUDENTS: IMPROVING THE STUDY.

Виконав: студент(ка) 2 курсу, групи КНТ-112м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

-

КУЗНЄЦОВ Т.В.

(ПРІЗВИЩЕ та ініціали)

Керівник КАПЛІСНКО Т.І.

(ПРІЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ

Кафедра програмних засобів

Ступінь вищої освіти магістр

Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КУЗНЄЦОВА Тараса Валерійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація інтелектуального сайту-чату для студентів: покращення навчального процесу. Research and software implementation of a Smart Chat Site for Students: Improving the study.

керівник проєкту (роботи) к.т.н, доцент, КАПЛІСНКО Тетяна Ігорівна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 1 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи 4. Керівництво програміста. 5. Керівництво оператора. 6. Тестування і результати. 7. Технічне завдання. 8. Текст програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-6 Основна частина	КАПЛІЄНКО Т.І., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання 8 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї	10-11 тижні	Додатки
8	Нормоконтроль та рецензування.	12 тиждень	
9	Захист роботи.	12 тиждень	

Студент

(підпис)

Тарас КУЗНІЦОВ

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Тетяна КАПЛІЄНКО

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи магістра:
152 с., 4 табл., 65 рис., 3 дод., 16 джерел.

ЧАТ, ВЕБЗАСТОСУНОК, JAVASCRIPT, TYPESCRIPT, C# РОЗКЛАД
ЗАКЛАДІВ ВИЩОЇ ОСВІТИ, ОБРОБКА ДАНИХ, ПОШУК СТУДЕНТІВ,
ЛИСТУВАННЯ.

Об'єкт дослідження – програмні засоби для реалізації інтелектуального чату та систем для покращення навчального процесу закладів вищої освіти (ЗВО).

Предмет дослідження – вебзастосунок інтелектуального чату для покращення навчального процесу ЗВО.

Мета роботи – створення вебзастосунку інтелектуального чату та додаткових систем, необхідних для покращення навчального процесу.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування JavaScript, мова програмування C#, мова програмування TypeScript, система для побудови клієнтських вебзастосунків, середовище розробки Visual Studio Code, Visual Studio персональний комп'ютер з процесором AMD Ryzen 5 під управлінням операційної системи Linux Ubuntu 20.04.

Результати. Створено програмний вебзастосунок, який виконує обробку даних, необхідних для покращення навчального процесу закладу вищої освіти.

Висновки. Розроблено інтелектуальний вебзастосунок для покращення навчального процесу закладів вищої освіти, необхідний для зручного користування студентами та викладачами університету.

Галузь використання – ЗВО.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 152 p., 4 tables, 65 figures, 3 appendix, 16 sources.

CHAT, WEB APPLICATION, JAVASCRIPT, TYPESCRIPT, C# SCHEDULE OF HIGHER EDUCATION INSTITUTIONS, DATA PROCESSING, STUDENT SEARCH, CORRESPONDENCE.

Object of research - software tools for the implementation of intelligent chat and systems to improve the educational process of higher education institutions (HEIs).

The subject of research is a web-based intelligent chat application to improve the educational process of higher education institutions.

The purpose of the study is to create a web-based intelligent chat application and additional systems necessary to improve the educational process.

Materials, methods and tools: structural and object-oriented programming, JavaScript programming language, C# programming language, TypeScript programming language, system for building client web applications, Visual Studio Code development environment, Visual Studio personal computer with AMD Ryzen 5 processor running Linux Ubuntu 20.04.

Results. A software web application has been created that processes data necessary to improve the educational process of a higher education institution.

Conclusions. An intelligent web application has been developed to improve the educational process of higher education institutions, which is necessary for convenient use by students and university teachers.

The field of application is higher education institutions.

ЗМІСТ

Вступ.....	9
1 Аналіз предметної області	10
1.1 Огляд існуючих аналогів та їх порівняльний аналіз	10
1.2 Огляд існуючих методів вирішення завдання	11
1.3 Огляд існуючих аналогів та їх порівняльний аналіз	11
1.4 Порівняльний аналіз аналогів.....	15
1.5 Постановка завдання до роботи	15
1.6 Підсумки аналізу існуючого стану проблеми.....	16
2 Вибір та обґрунтування структури системи, яка проєктується	18
2.1 Вибір інструментарію розробки програмного продукту	18
2.2 Вибір середовища проєктування клієнтської частини.....	23
2.3 Вибір середовищ для швидкого розгортання застосунку.....	26
2.4 Вибір бази даних	26
2.5 Вибір середовища розробки.....	29
2.6 Архітектура вебзастосунку	30
2.7 Вимоги до застосунку, що розробляється	33
2.8 Висновки до розділу	34
3 Основні рішення щодо реалізації компонентів системи	35
3.1 Алгоритм функціонування програми	35
3.2 Основні рішення щодо реалізації компонентів системи	37
3.2.4 Основні рішення щодо створення інтелектуального чату.....	44
3.3 Підсумки опису програми	45
4 Керівництво програміста.....	46
4.1 Призначення та умови застосування програми	46
4.2 Характеристики програми.....	47
4.3 Звернення до програми.....	49
4.4 Вхідні та вихідні дані.....	49
4.5 Повідомлення програмісту.....	49

5 Керівництво оператора	51
5.1 Призначення та умови застосування програми	51
5.2 Виконання програми.....	52
5.3 Висновки по розділу	67
6 Тестування і результати	68
6.1 Опис методики проведення тестування.....	68
6.2 Результати тестування	68
Висновки	76
Перелік джерел посилань	78
Додаток А Технічне завдання.....	80
Додаток Б Текст програми	86
Додаток В Слайди презентації.....	144

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

JS	– JavaScript;
SPA	– Single page application;
TS	– TypeScript;
SSR	– Server side rendering;
БД	– база даних;
ЗВО	– заклад вищої освіти;
НУ	– національний університет;
ПЗ	– програмне забезпечення;
ПП	– програмний продукт;
СУБД	– системи управління базами даних.

ВСТУП

У сучасному університетському освітньому середовищі вступає новий етап розвитку, в якому технології є не лише допоміжним інструментом, але й важливим чинником трансформації навчального процесу. У цьому контексті питання доступності та ефективності інформації набувають особливого значення для студентів, які перебувають у центрі навчальної діяльності.

Освітній процес стає все більш насиченим технологічними інноваціями, а разом з ним актуалізується потреба у зручних та ефективних інструментах. Однією з головних проблем, з якою стикаються студенти, є складність управління та інформування про свій навчальний план. Традиційні паперові розклади вже не можуть впоратися з класичними методами організації, оскільки потік інформації занадто великий.

Для подолання цих викликів у цьому дипломному проєкті розглядаються дослідження та розробка інтелектуального чат-сайту для студентів. Цей інноваційний інструмент поєднує найсучасніші технології з розумним підходом і має на меті оптимізувати взаємодію між студентами та навчальним процесом.

Забезпечуючи легкий доступ до розкладу та підтримуючи ефективну організацію навчання, інтелектуальний чат-сайт вирішує виклики, з якими університети зіткнулися в новому десятилітті. Сайт має на меті вдосконалити навчальний процес шляхом створення інноваційного середовища для спілкування, обміну інформацією та подальшого розвитку студентської спільноти, а також допомогти студенту якнайшвидше дізнаватися та моніторити різні аспекти його студентського життя.

Ця робота висвітлює шляхи розробки, як розробити та впровадити інтелектуальний чат-сайт, який не тільки адаптується до потреб сучасних студентів, але й сприяє створенню більш інтерактивного та захоплюючого освітнього середовища.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд існуючих аналогів та їх порівняльний аналіз

Перш ніж аналізувати предметну область, необхідно визначити проблему, що розглядається. В даному випадку такою проблемою є необхідність розробки інтелектуального сайту-чату для покращення навчального процесу, тобто цей інструмент потрібен викладачам та студентам.

Переходячи безпосередньо до аналізу, необхідно детально проаналізувати процес листування та моніторингу різних функцій ЗВО. Будучи знайомими зі звичками роботи та методологією подібних вебзастосунків, можемо легко перерахувати поточні потреби в програмному забезпеченні, яке вирішує наступні проблеми:

- зручна авторизація користувача;
- можливість створення чатів;
- зручний перегляд користувачів за групами;
- зручний пошук розкладу групи за її номером;
- сортування груп адміністратором;
- зручний перегляд розкладу користувача;
- можливість створення розкладу по групам;
- можливість скривати/відкривати дані, якими користувач хоче поділитися;
- можливість сповіщення користувачів при зміні важливих даних;
- можливість відправлення текстового повідомлення користувачам;
- зручний інтерфейс застосунку повинен бути розроблений англійською та українською мовами;

Рішення перерахованих вище завдань, а саме часткова автоматизація повсякденних справ, значно поліпшує навчальний процес і перегляд програм за допомогою спеціальних фільтрів.

1.2 Огляд існуючих методів вирішення завдання

Традиційний метод відстеження розкладу в ЗВО залишається застарілим, що затримує процес поширення важливої інформації між студентами та викладачами.

Новий вебсайт пропонує широкий спектр функцій і зручний інтерфейс, що робить його ефективним інструментом для управління процесом навчання. Схваливши користувача, студенти та викладачі можуть отримати доступ до основних функцій інтелектуального сайту-чату.

Особлива увага приділяється системі сповіщень, яка швидко повідомляє користувачів про зміни важливих даних та новин. Простота спілкування забезпечується можливістю відправки текстових повідомлень всередині платформи.

1.3 Огляд існуючих аналогів та їх порівняльний аналіз

1.3.1 Система «Blackboard Learn»

Офіційний сайт програми «Blackboard Learn»: <https://www.blackboard.com/>

Даний програмний продукт (ПП), представлений на рис. 1.1 [1], призначено для зручності навчання.

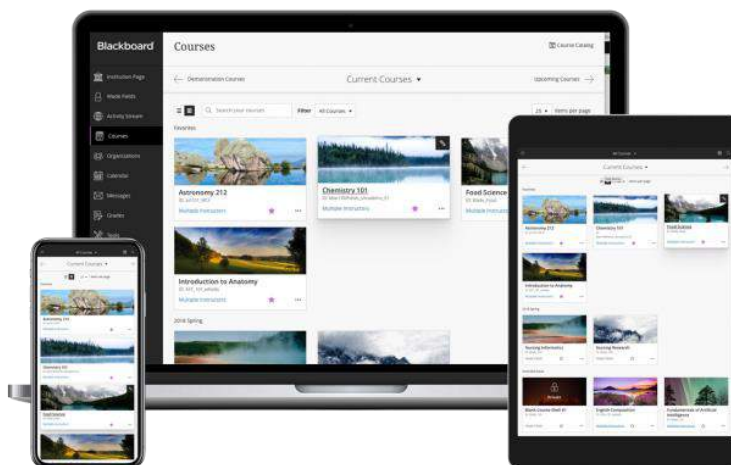


Рисунок 1.1 – Система управління навчанням «Blackboard Learn» [1]

Його базовий функціонал:

- створення та редагування курсів;
- додавання навчального матеріалу, завдань та інших ресурсів;
- управління розкладом та темами занять;
- форуми для обговорення тем та завдань;

Переваги даного ПП:

- багатоплатформність;
- багатомовність;
- багато матеріалів.

Його недоліки:

- незручний та перевантажений дизайн сайту;
- система p-2-b (person to business);
- неможливо додати своє ЗВО;
- відсутня можливість безкоштовного використання.

1.3.2 Система «Schoology»

Офіційний сайт програми «Schoology»: <https://www.powerschool.com/>

Даний програмний продукт, представлений на рис. 1.2 [2], призначено діджиталізації та покращення навчального процесу.

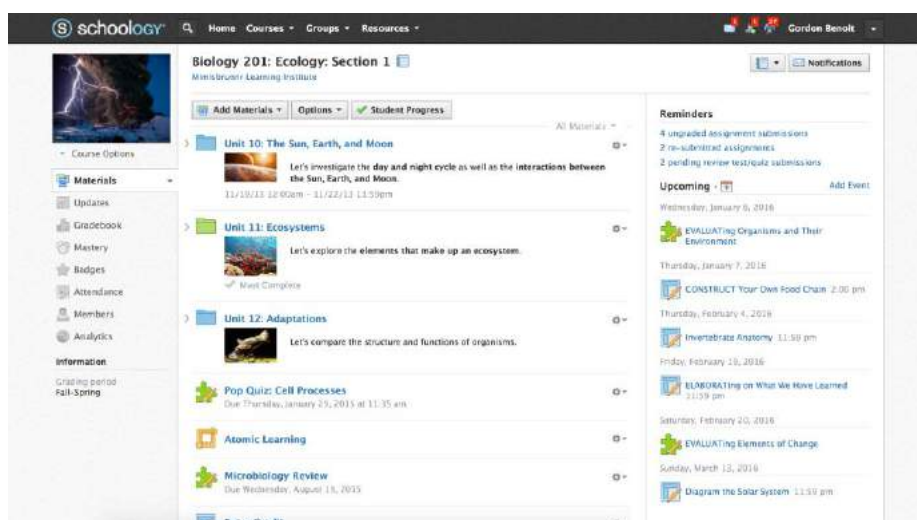


Рисунок 1.2 – Система управління навчанням Schoology [2]

Його базовий функціонал:

- ведення календаря подій та розкладу занять;
- відстеження успішності студентів та інші аналітичні засоби для оцінки ефективності навчання;
- можливість надання зворотного зв'язку.

Переваги даного ПП:

- всеохопний функціонал;
- багатоплатформність;
- захист доступу до курсів та забезпечення конфіденційності інформації;
- онлайн підтримку для користувачів.

Його недоліки:

- відсутність простого додавання студентських груп;
- орієнтовано під шкільні заклади освіти;
- відсутність простих маніпуляцій з розкладом та студентами;
- відсутність вибору різних мов;
- відсутність пошуку аудиторії.

1.3.3 Система «Edmodo»

Офіційний сайт програми «Edmodo»: <https://www.edmodo.com/>

Даний ПП, представлений на рис. 1.3 [3], призначено для спрощення комунікації та співпраці між учнями, викладачами та батьками.

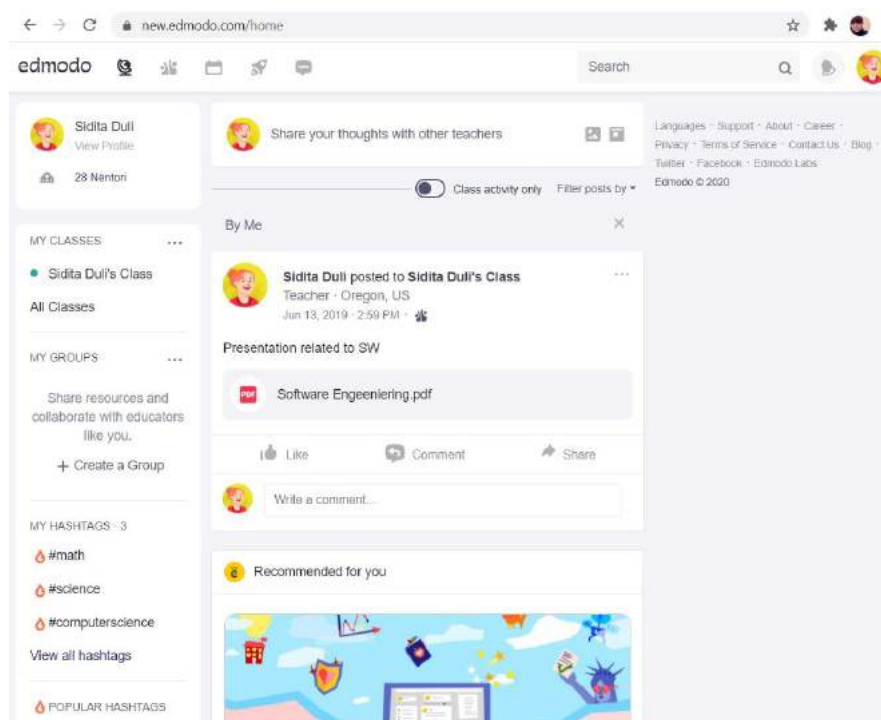


Рисунок 1.3 – Система управління навчанням Edmodo [3]

Його базовий функціонал:

- виставлення та відстеження оцінок за завдання та інші активності;
- ведення календаря подій та завдань для учнів та викладачів;
- забезпечення безпеки доступу та конфіденційності інформації.

Переваги даного ПП:

- всеохопний функціонал;
- багатоплатформність;
- можливість відстеження оцінок;
- гарний та зручний інтерфейс;
- онлайн підтримку для користувачів.

Його недоліки:

- відсутність пошуку аудиторій за їх номером;
- відсутність нумерації пар;
- відсутність пошуку пар для викладача.

1.4 Порівняльний аналіз аналогів

Проаналізувавши готові рішення, описані в попередньому розділі, було виконано їх порівняння з точки зору функціональності, плюсів і мінусів. Результати (таблиця 1.1) доводять, що всі готові рішення мають багато переваг і недоліків. Це показує важливість написання власного вебзастосунку.

Таблиця 1.1 – Результати порівняння

	Blackboard Learn	Schoology	Edmodo
Зручний інтерфейс	+	-	-
Можливість зручного спілкування з викладачами	+	—	+
Можливість предметів за групами	—	+	-
Можливість пошуку аудиторій	-	-	-
Багатомовність	-	-	+
Перегляд розкладу наперед	+	+	+

1.5 Постановка завдання до роботи

Після проведеного аналізу готових рішень було формалізовано завдання до роботи, що полягає у створенні клієнт-серверного застосунку (оскільки розроблюване програмне забезпечення має надати засоби отримання даних з централізованого сховища даних та надавати інтерфейс для роботи з нею), що відповідає наступним критеріям:

- надає можливість зручної авторизація користувача;
- надає можливість зручної реєстрації користувача;
- надає можливість створення чатів;

- надає зручний перегляд користувачів за групами;
- надає зручний перегляд розкладу, предметів, аудиторій, груп та основної інформації користувачів;
- надає можливість створення розкладу по групам;
- надає можливість скривати/відкривати дані, якими користувач хоче поділитися;
- надає можливість сповіщення користувачів при зміні важливих даних;
- надає можливість відправлення текстового повідомлення користувачам;
- надає зручний мультимовний інтерфейс застосунку;
- має зручний користувацький інтерфейс, що відповідає стандартам проектування UI/UX-інтерфейсів, а також Material Design;
- є кросплатформним - може бути використано на UNIX-подібних операційних системах, таких як Linux, MacOS, Android, iOS, а також на ОС Windows.

На вході програма повинна отримувати дані, за якими потрібно здійснити пошук, а також інформацію про студентів, викладачів, груп, аудиторій, а також можливість листування.

Оцінка успішності розробленого програмного забезпечення проводиться шляхом порівняння його на відповідність критеріям, зазначеним у пункті 1.3 пояснювальної записки.

1.6 Підсумки аналізу існуючого стану проблеми

В кінці аналізу предметної області, крім огляду доступних рішень проблеми, можемо зробити наступні висновки:

- більшість існуючого програмного забезпечення представляє собою комплексні рішення вебзастосунків для покращення навчального процесу;
- існуюче програмне забезпечення не орієнтовано на ЗВО саме за певними критеріями, а в більшість випадків створене для бізнес навчання;

– розглянуте програмне забезпечення (що є найбільш використовуваним і популярним) має застарілий дизайн і архітектуру побудови графічного інтерфейсу користувача (англ «Graphical User Interface», скорочено «GUI»).

Після перевірки результатів можна підвести підсумки. В даний час дуже важливо створити сучасну систему професійних інтелектуальних чат-сайтів для покращення навчального процесу в середніх школах із сучасним дизайном та певним набором параметрів для відображення програми відповідно до певних критеріїв, таких як вчителі, групи чи аудиторія.

За результатами аналізу узагальнені технічні завдання перераховані в додатку А пояснювальної записки проєкту.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного продукту

Важливим етапом життєвого циклу системи моніторингу, особливо при розробці ПП, є вибір інструментів і підключення необхідної технології до проєкту.

Основним завданням програмного забезпечення є вдосконалення навчального процесу шляхом впровадження інтелектуальних чатів для студентів і викладачів та моніторингу різних аспектів університетської системи.

Розроблюваний ПП повинен обов'язково містити такі модулі:

- виведення та обробка даних;
- модуль зв'язку серверу з графічною частиною.

Для проєктування ПП були обрані такі засоби, як мова програмування та середовище розробки (Integrated Development Environment – IDE).

2.1.1 Вибір мови програмування

Для вирішення завдання вдосконалення навчального процесу за допомогою інтелектуального чату для студентів і викладачів та моніторингу різних аспектів діяльності ЗВО необхідно розглянути найсучасніші мови програмування для реалізації програмного забезпечення. Нижче розглядаються найпопулярніші мови програмування для розробки вебзастосунку, їх переваги та недоліки.

JavaScript (JS) — динамічна, об'єктно-орієнтована мова програмування прототипів; реалізація стандарту ECMAScript. Це дозволяє клієнтській стороні взаємодіяти з користувачем, керувати браузером, асинхронно спілкуватися з сервером і змінювати структуру і зовнішній вигляд вебсторінки [4].

TypeScript – мова програмування, представлена Microsoft в 2012 році і позиціонується як засіб розробки вебзастосунків, що розширює можливості JavaScript [5].

TypeScript з'явився на світ через усвідомлення недоліків JavaScript у великомасштабних застосунках як всередині Microsoft, так і іншими користувачами JavaScript. Проблеми з розробкою складних JavaScript-застосунків призвели до необхідності спрощення розробки компонентів мови.

Розробники TypeScript шукали рішення, яке не погіршило б сумісність стандартів і крос-платформну підтримку; вони знали, що тільки стандарт ECMAScript підтримуватиме програмування на основі класів у майбутньому, і TypeScript ґрунтувався на цьому припущенні і був розроблений на цій основі. Результатом став компілятор JavaScript, що працює на основі набору розширень синтаксичної мови, і пропозиція щодо перетворення цих розширень у JavaScript. У цьому сенсі TypeScript представляє те, що очікується від ECMAScript. Унікальним аспектом мови є додавання статичної типізації до TypeScript, що дозволяє статично аналізувати мову і полегшує підтримку інструментів та IDE.

JavaScript - це об'єктно-орієнтована мова, але вона відрізняється від традиційних класово-орієнтованих мов способом роботи з об'єктами. Крім того, JavaScript має багато особливостей, характерних для функціональних мов, таких як функції як першокласні об'єкти, об'єкти як списки, фігурні дужки, анонімні функції та закриття, що надає мові додаткової гнучкості.

Незважаючи на синтаксис, схожий на C, JavaScript відрізняється від C деякими фундаментальними особливостями:

- об'єкти з можливістю інтроспекції;
- функції як об'єкти першого класу;
- автоматичне приведення типів;
- автоматичне прибирання сміття;
- анонімні функції.

У мові бракує таких можливостей:

- стандартні інтерфейси до вебсерверів і баз даних;

- стандартні бібліотеки - зокрема, відсутні програмні інтерфейси для застосунків, що працюють з файловою системою, керують потоками вводу/виводу та основними бінарними типами даних;

- системи керування пакетами, які відстежують залежності та встановлюють їх автоматично.

C# – це об'єктно-орієнтована мова програмування із захищеною системою сценаріїв для платформи .NET. Розроблена Андерсом Гейлсбергом, Скоттом Вілтанутом і Пітером Голде під егідою Microsoft Research (частина Microsoft), C# за синтаксисом схожа на C++ і Java. Вона має строгу статичну типізацію і підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класу, атрибути, події, властивості, винятки і коментарі в стилі XML C# багато запозичила з C++, C, Object Pascal, Modules і Smalltalk, і усунула багато проблемних шаблонів при розробці програмних систем, заснованих на їх реалізаціях.

Python – багатопарадигмальна мова програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване, цільове та функціональне програмування. Узагальнені задачі програмування вирішуються за допомогою динамічної типізації. Аспектно-орієнтоване програмування частково підтримується декораторами, а додаткові фреймворки забезпечують більш повну підтримку. Такі методи, як контрактне та логічне програмування, можуть бути реалізовані за допомогою бібліотек та розширених бібліотек. Основні архітектурні особливості включають динамічний запис, автоматичне управління пам'яттю, повне самоаналіз, механізми обробки винятків, підтримку багатопотокових обчислень із глобальним декомутативним блокуванням (GIL) та структури даних високого рівня. Мова надає можливість розділення програми на модулі та об'єднувати їх в один пакет [6].

RНР – це скриптова мова загального призначення, яка широко використовується для розробки вебзастосунків. Наразі вона підтримується більшістю хостинг-провайдерів і є однією з основних мов, що використовуються для створення динамічних вебзастосунків.

У сфері вебпрограмування, особливо на стороні сервера, PHP є однією з найпопулярніших мов сценаріїв (поряд з JSP, Perl і мовами, що використовуються в ASP.NET).

Популярність PHP у сфері розробки вебзастосунків визначається наявністю великої кількості вбудованих інструментів і додаткових модулів для розробки вебзастосунків. Основними з них є наступні:

- автоматичне вилучення POST- і GET-параметрів, а також змінних оточення вебсервера в зумовлені масиви;
- взаємодія з великою кількістю різних систем управління базами даних через додаткові модулі;
- автоматизована відправка HTTP-заголовків;
- робота з HTTP-авторизацією;
- робота з куками і сесіями;
- робота з локальними і віддаленими файлами, сокетамі;
- обробка файлів, що завантажуються на сервер;
- робота з Xforms.

До 2010-х років PHP використовували сотні тисяч розробників, а станом на травень 2016 року, згідно з рейтингом пошукових систем корпорації ТІОВЕ, PHP посідає 6-те місце серед мов програмування [7].

Таким чином, кожна з вищезазначених мов підходить для реалізації цього проекту та має низку корисних функцій, які є дуже корисними для впровадження моніторингу розкладу в навчальних закладах. Завдяки вищезазначеним мовам програмування здійснюється візуалізація, обробка та аналіз даних. Кожна з вищезазначених мов розвивається з кожним роком і тому надає чудову можливість для швидкої реалізації речей, які раніше займали багато часу.

При аналізі різних мов програмування були визначені порівняльні характеристики вищезазначених мов програмування, які наведені у таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика мов програмування

Критерії порівняння	JavaScript	TypeScript	PHP	Python	C#
Мова створювалася спеціально для аналізу даних	+	—	—	+	+
Наявність динамічної типізації	+	++	+	+	+
Можливість статичної типізації	-	+	-	-	++
Наявність зрозумілої та повної документації	+	+	+	+	++
Можливість розробки серверної частини	+	+	+	+	++
Можливість розробки клієнтської частини	+	+	-	-	+
Продуктивність роботи	+	+	+	+-	++
Можливість швидкого підключення сторонніх бібліотек	++	+-	+	+	++
Наявність великої кількості наборів сторонніх пакетів	++	—	+	—	++
Назва пакетного менеджера	NPM	NPM	Composer	Pip	Nuget
Взаємодіє з іншими мовами	+	+	+	+	+

Як видно з таблиці 2.1, TypeScript (TS), JavaScript (JS) та C# є трьома найпопулярнішими мовами програмування. Для мови JS написана низка бібліотек, які можна встановити за допомогою менеджера пакетів NPM. TS є мовою програмування JS і використовується для імітації статичної типізації, що полегшує написання коду. JS підтримується всіма сучасними браузерами. Мова C# має багату спільноту і багато бібліотек написано для роботи на серверній стороні вебзастосунку, вона також є дуже швидкою. Важливо зазначити, що мову

програмування TS слід використовувати лише на стороні клієнта, тоді як C# - на стороні сервера.

Таким чином, TS і C# є кращими мовами для розробки програмного забезпечення.

2.2 Вибір середовища проєктування клієнтської частини

Для того, щоб спростити реалізацію вебзастосунку на стороні клієнта, необхідно використовувати фреймворк для швидкої та легкої побудови користувацьких інтерфейсів. Тому було порівняно чотири популярні фреймворки та обрано один з них для вирішення завдання створення користувацького інтерфейсу для програми моніторингу програм.

Nuxt.js – це фреймворк для вебзастосунків, побудований на Vue.js, JavaScript-фреймворку, який широко використовується для написання користувацьких інтерфейсів. Він має вбудовані функції, такі як рендеринг на стороні сервера (SSR), статичний рендеринг сайту (SSG) та прогресивні вебзастосунки (PWA). Nuxt.js найкраще підходить для вебпроєктів, тому SEO-оптимізація, швидке завантаження, дружній інтерфейс є пріоритетом, і його часто використовують для створення вебсайтів, керованих контентом, платформ електронної комерції та інших проєктів.

Vue – це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від багатьох монолітних фреймворків, Vue розроблений для поступового впровадження. Його ядро полегшує інтеграцію з іншими бібліотеками та існуючими проєктами, вирішуючи проблеми в першу чергу на рівні подання. З іншого боку, Vue також добре підходить для створення складних односторінкових застосунків (SPA, Single-Page Applications) [8].

React – бібліотека JavaScript з відкритим вихідним кодом для розробки користувацьких інтерфейсів, розробляється і підтримується Facebook, Instagram та спільнотою індивідуальних розробників і компаній React можна використовувати для розробки односторінкових і мобільних застосунків. Його мета - забезпечити

швидкість, простоту і масштабованість. Як бібліотека для розробки користувацького інтерфейсу, React часто використовується в поєднанні з іншими бібліотеками, такими як MobX, Redux та GraphQL [9].

Angular – фреймворк від Google для створення клієнтських застосунків. Насамперед, він спрямований на розробку SPA-рішень (Single Page Application), тобто односторінкових застосунків. В цьому відношенні Angular є наступником іншого фреймворку AngularJS. У той же час, Angular - це не нова версія AngularJS, а принципово новий фреймворк: він надає такі можливості, як двонаправлене зв'язування (зміна даних моделі в одній частині інтерфейсу може динамічно змінювати дані в іншій частині), шаблони, маршрутизація та інші функції [10].

SSR (Server-Side Rendering) – рендеринг клієнтських або універсальних застосунків у вигляді HTML на сервері. SSR (Server-Side Rendering - рендеринг на стороні сервера) рендерить повний HTML-код сторінки на сервері у відповідь на запит. Це відбувається до того, як браузер отримає відповідь, таким чином заповнюючи шаблон на стороні клієнта і уникаючи додаткових запитів даних. Виконуючи логіку сторінки та рендеринг на сервері, він уникає надсилання клієнту занадто великого обсягу JavaScript і допомагає досягти швидкого часу взаємодії (TTI). Це має сенс, оскільки SSR надсилає лише текст і посилання до браузера користувача. Такий підхід добре працює на різних пристроях і з різною якістю мережі, а також дозволяє проводити цікаві оптимізації браузера, наприклад, потокове передавання документів. Завдяки SSR користувачам не потрібно чекати на відображення відповідного прив'язаного до процесора JavaScript, перш ніж користуватися сайтом. Навіть якщо використання стороннього JS неминуче, використання SSR для зменшення навантаження на JS може збільшити "бюджет" для інших процесів. Однак такий підхід має суттєвий недолік. Створення сторінок на сервері займає багато часу і часто призводить до повільного часу до першого байта (TTFB).

Односторінкові застосунки (SPA) - це вебзастосунки або вебсайти, які взаємодіють з користувачем шляхом динамічного переписування існуючої

вебсторінки з новими даними з вебсервера, замість того, щоб веббраузер завантажував всю нову сторінку за замовчуванням. Метою є пришвидшення переходів, що робить вебсайт схожим на нативний застосунок.

Односторінкові застосунки можуть імітувати роботу десктопних застосунків. Архітектура розроблена таким чином, що при переході на нову сторінку оновлюється лише частина контенту. Таким чином, ті самі елементи не потрібно перезавантажувати. Це дуже зручно як для розробників, так і для користувачів.

У SPA весь необхідний код HTML, JavaScript і CSS завантажується браузером або за одне завантаження сторінки, а відповідні ресурси динамічно завантажуються і додаються на сторінку в міру необхідності. Для розпізнавання та навігації окремими логічними сторінками в застосунку можна використовувати хешування місцезнаходження або API історії HTML5, але при цьому сторінки не перезавантажуються і управління не передається іншій сторінці в будь-який момент процесу [11].

Під час аналізу різних графічних фреймворків було створено порівняльну характеристику вищесказаних фреймворків, яка наведено у таблиці 2.2.

Таблиця 2.2 – Порівняльна характеристика графічних фреймворків

Фреймворки	React	Vue.js	Angular	Nuxt.js
Низький поріг входження	-	+	—	+—
Підтримка TypeScript	+	++	+	++
Підтримка SPA	+	+	+	+
Наявність зрозумілої та повної документації	+	+	+	++
Легке підключення графічних фреймворків	-	+	-	++
Легке підключення сторонніх бібліотек	+	+	-	++
Продуктивність роботи	+	++	+	++
Управління станами	+	+	+	+
Підтримка SSR	-	-	+	+

Як видно з таблиці 2.2 найбільші переваги має SSR фрейворк Nuxt.js який базується на SPA фреймворці Vue.js, оскільки він надає багато корисних функцій з коробки. Фреймворк Nuxt.js пропонує багато можливостей для збільшення продуктивності і легкості використання. Слід зазначити, що після завантаження першої сторінки сайту архітектура вебзастосунку перемикається з SSR на SPA, що допомагає вебзастосунку працювати як єдине ціле і переходити на різні сторінки сайту без необхідності перезавантаження сторінки, що називається маршрутом.

Отже, підбиваючи вищесказане для розробки ПП було обрано фреймворк Nuxt.js.

2.3 Вибір середовищ для швидкого розгортання застосунку

Docker - це відкрита платформа для розробки, доставки та запуску застосунків. Docker дозволяє відокремити застосунки від інфраструктури, щоб швидко доставляти програмне забезпечення. За допомогою Docker можна керувати інфраструктурою так само, як і своїми застосунками. Використовуючи переваги для доставки, тестування та розгортання коду, ви можете значно скоротити затримку між написанням коду та його запуском у виробництво.

Однією з переваг технології Docker є те, що він не залежить від платформи. Це означає, що можна написати спосіб запуску програми на Windows і без проблем запустити її на MacOS. Це дає можливість дуже швидко переносити та налаштовувати вебзастосунки на різних серверах.

2.4 Вибір бази даних

Розроблюваний ПП повинен обов'язково містити такі модулі:

- зберігання даних;
- розрахунковий;
- виводу даних;

- модуль зв'язку всіх елементів ПП.

Розроблене програмне забезпечення використовує комбінацію чату ЗВО, розкладу, даних викладачів, груп та студентів, тобто групові заняття, розклад, чат, повідомлення в чаті та викладачів. Ці дані зберігаються в наступних сховищах:

- текстові файли (CSV, txt, xml,...);
- бінарні файли даних;
- не реляційні бази даних (NoSQL сховища, MongoDB);
- реляційні бази даних (MySQL, MS SQL Server, Firebird, тощо).

Обраний тип сховища повинен забезпечувати достатню гнучкість формату, швидкий пошук даних, можливість зберігати великі обсяги даних і розширювати функціональність. Усім цим вимогам відповідає реляційна система управління базами даних (СУБД). Переваги використання реляційної СУБД:

- простота розуміння користувачем. Єдина інформаційна структура - "таблиці";
- жорсткі правила проєктування, засновані на математичному апараті;
- повна незалежність даних;
- для написання прикладного програмного забезпечення і організації запитів не обов'язково знати конкретну конфігурацію бази даних у зовнішній пам'яті.

SQLite – найбільш широко використовуваний механізм баз даних SQLite у світі вбудований у всі мобільні телефони та більшість комп'ютерів, а також входить до складу багатьох інших застосунків, якими люди користуються щодня. Формат файлів SQLite стабільний, перехресні файли баз даних SQLite широко використовуються як контейнери для передачі контенту між системами та як формат даних для довготривалого зберігання.

Knex.js – це конструктор SQL запитів для Postgres, MSSQL, MySQL, MariaDB, SQLite3, Oracle та Amazon Redshift, розроблений для гнучкості, портативності та зручності використання. Обіцяний інтерфейс для традиційних зворотних викликів у стилі вузла і більш чистого асинхронного управління потоками, потоковий інтерфейс, повнофункціональний конструктор запитів і

схем, підтримка транзакцій (з точками збереження), злиття між різними клієнтами запитів і діалектами [12].

MySQL – це безкоштовна реляційна система управління базами даних, розроблена компанією TCX для підвищення продуктивності транзакцій у великих базах даних. Ця СУБД з відкритим вихідним кодом була створена як альтернатива комерційним системам. Спочатку MySQL була дуже схожа на mSQL, але з часом її було розширено, і зараз MySQL є однією з найпопулярніших систем управління базами даних. Вона в основному використовується для створення динамічних вебсторінок завдяки чудовій підтримці різних мов програмування [13].

Microsoft SQL Server - це власна система управління реляційними базами даних, розроблена компанією Microsoft. Як сервер баз даних, це програмний продукт, основною функцією якого є зберігання та пошук даних за запитом інших програмних додатків, які можуть працювати як на тому ж комп'ютері, так і на іншому комп'ютері в мережі (включаючи Інтернет) [14].

Microsoft SQL Server Management Studio (SSMS) - це програмне забезпечення, розроблене компанією Microsoft, яке використовується для конфігурування, управління та адміністрування всіх компонентів Microsoft SQL Server. Вперше запущена з Microsoft SQL Server 2005, вона є наступником Enterprise Manager в SQL 2000 або більш ранніх версіях. Інструмент включає в себе як редактори сценаріїв, так і графічні інструменти, які працюють з об'єктами і функціями сервера.

Entity Framework - це набір технологій ADO.NET, які підтримують розробку програмних застосунків, керованих даними. Архітектори та розробники застосунків на основі даних часто стикаються з необхідністю досягнення двох дуже різних цілей. Їм потрібно моделювати сутності, взаємозв'язки і логіку бізнес-проблеми, яку вони намагаються вирішити, а також працювати з механізмом даних, який використовується для зберігання і вилучення даних. Дані можуть охоплювати декілька систем зберігання, кожна з яких має власні протоколи. Навіть застосунки, що працюють на одній системі зберігання даних, повинні

балансувати між вимогами системи зберігання даних і вимогами написання ефективного і зручного для підтримки коду програми [15].

Отже, підсумовуючи вищесказане у якості системи зберігання даних було обрано MsSql, у якості утиліти для конфігурування бази даних було обрано SSMS, а також у якості фреймворка для роботи над базою даних на стороні сервера був обраний фреймворк Entity Framework.

2.5 Вибір середовища розробки

Для того, щоб програмування на вибраній мові програмування було швидке та зручне було обране спеціальне середовище розробки, для розробки клієнтської сторони було обране таке середовище як Visual Studio Code, а для розробки серверної частини проєкту було обране середовище Visual Studio.

IDE Microsoft Visual Studio – це інтегроване середовище розробки програмного забезпечення від Microsoft і повноцінне середовище розробки програмного забезпечення для всіх підтримуваних платформ, включаючи Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Compact Framework, .NET Compact Framework і Windows Forms для технології Microsoft Silverlight, а також інтегроване середовище розробки програмного забезпечення для всіх підтримуваних платформ, включаючи підтримку консольних і графічних застосунків, вебсайтів і багатьох інших інструментів, вебзастосунки і вебслужби можна розробляти як з використанням власного, так і керованого коду [16].

IDE Visual Studio Code, також відомий як VS Code - це редактор вихідного коду, розроблений компанією Microsoft для Windows, Linux та macOS. Функції включають підтримку налагодження, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш, налаштування та встановлювати розширення, які додають функціональності.

2.6 Архітектура вебзастосунку

В якості архітектури вебзастосунку була обрана SSR архітектура.

SSR – це збірний термін для позначення набору технологій, які дозволяють вебзастосункам, таким як сервіс Google Gmail, працювати як одна вебсторінка у веббраузері. З точки зору користувача, ця технологія не вимагає повного або часткового перезавантаження вебсторінки з сервера, оскільки всі візуальні елементи відображаються безпосередньо в браузері за допомогою JavaScript шляхом маніпулювання DOM (структурою документа), а швидкість, з якою інтерфейс реагує на дії, є дуже привабливою. Слід зауважити, що після завантаження першої сторінки сайту, архітектура вебзастосунку переходить від SSR до SPA, це допомагає вебзастосунку працювати як одне ціле, та без перезавантаження сторінок переходити по різних сторінкам сайту, так званим роутам.

Таким чином, вебзастосунки стають дуже схожими на звичайні програми для робочих станцій, які завантажують інформацію з Інтернету, але середовищем їх виконання є браузер, а не операційна система, і в результаті їм доводиться забезпечувати всі накладні витрати, пов'язані з виконанням стороннього коду, такі як управління пам'яттю, забезпечення безпечного середовища і взаємодія з системними функціями та апаратним середовищем.

Представлення займають центральне місце в архітектурі SPA. Результатом представлення є звичайнісінький HTML, який відображається браузером. На відміну від "транзитивних" WEB 2.0 застосунків, які активно маніпулюють DOM-структурою документа, наприклад, за допомогою jQuery або підкреслення, SPA-застосунки використовують DOM тільки для запису змін, а не для їх читання, для відстеження змін у DOM використовують моделі.

Модель - це набір даних, функцій для маніпулювання даними та подій. Всі дані в моделі зберігаються в пам'яті. Щоб гарантувати, що дані в моделі і дані, які відображаються в поданні, не будуть пошкоджені, подання підписується на події моделі і відстежує зміни даних в моделі. Модель також відповідає на

повідомлення від представлення і виконує запити на отримання або відправку даних (зокрема, з використанням методології REST), забезпечуючи таким чином інтегральний зв'язок між вебзастосунком і сервером.

Принцип роботи SSR фреймворку NuxtJs показано у блок-схемі на рис. 2.1.

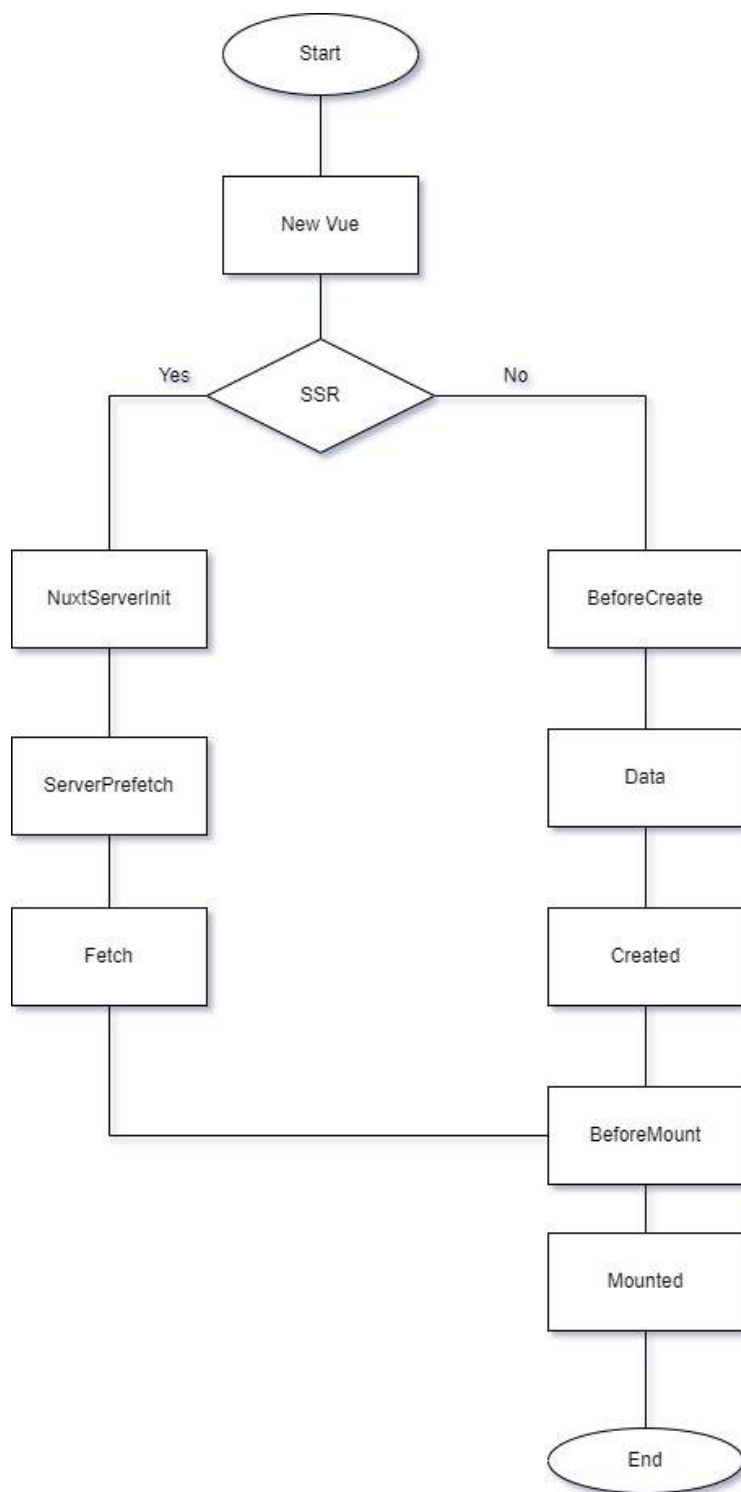


Рисунок 2.1 – Принцип роботи SPA фреймворку Vue.js

Нижче наведено хуки життєвого циклу фреймворку NuxtJs та їхні описи:

- спочатку відбувається ініціалізація самого фреймворку за допомогою `new Vue()`;
- `BeforeCreate` – одразу після запуску екземпляра дані та події викликаються синхронно до того, як вони будуть спостерігатися, і до того, як всі події та дані будуть ініціалізовані;
- `Created` – викликається синхронно після створення екземпляра. На цей момент екземпляр завершив параметризацію. Тобто створено спостереження даних, обчислювані властивості, методи та виклики спостережень/подій. Однак, фаза побудови ще не розпочалася і властивість `$el` ще не доступна;
- `BeforeMount` – викликається безпосередньо перед початком рендерингу: функція рендеру викликається вперше;
- `Mounted` – викликається коли екземпляр вже змонтовано, елемент, переданий до `app.mount`, замінюється на новостворений `vm.$el`. Якщо кореневий екземпляр змонтовано до елемента у документі, `vm.$el` також буде присутній у документі під час виклику змонтованого екземпляра.

Потім, коли дані змінюються, змінені компоненти перебудовуються.

У якості архітектури застосунку було обрано REST API.

REST, або REpresentational State Transfer, - це архітектурний стиль для забезпечення стандартів між комп'ютерними системами в Інтернеті, що полегшує взаємодію систем між собою. REST-сумісні системи, які часто називають RESTful-системами, характеризуються тим, що вони є бездержавними і розділяють проблеми клієнта і сервера. В архітектурному стилі REST реалізація клієнта і реалізація сервера можуть бути виконані незалежно, не знаючи один про одного. Це означає, що код на стороні клієнта може бути змінений в будь-який час без впливу на роботу сервера, а код на стороні сервера може бути змінений без впливу на роботу клієнта. Поки кожна сторона знає, в якому форматі надсилати повідомлення іншій стороні, вони можуть залишатися модульними та відокремленими. Відокремлюючи проблеми користувацького інтерфейсу від проблем зберігання даних, покращується гнучкість інтерфейсу на різних

платформах та покращуємо масштабованість за рахунок спрощення серверних компонентів. Крім того, розділення дозволяє кожному компоненту розвиватися незалежно.

Для вебслужби, побудованої на REST застосовується термін «RESTful».

Принцип роботи REST показано на рисунку 2.2.

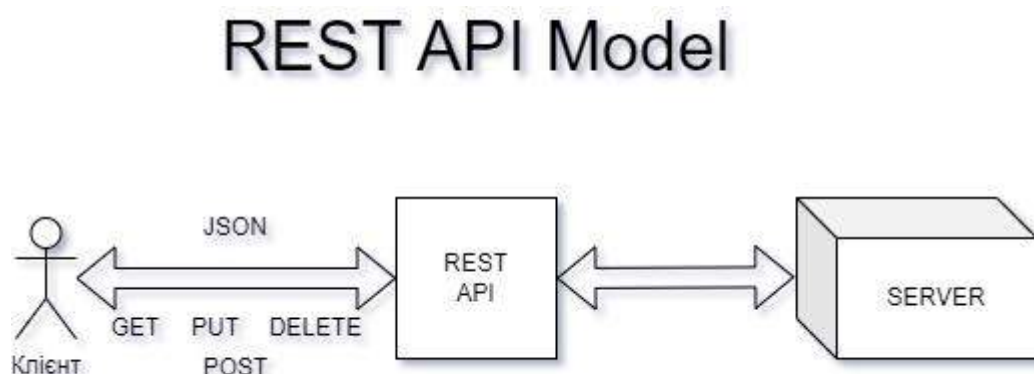


Рисунок 2.2 – Принцип роботи REST API застосунку

2.7 Вимоги до застосунку, що розробляється

Даний програмний продукт повинен мати такий функціонал, як:

- авторизація користувача;
- можливість створення чатів;
- можливість перегляду користувачів за групами;
- пошук розкладу групи за її номером;
- сортування груп адміністратором;
- перегляд розкладу користувача;
- можливість створення розкладу по групам;
- можливість скривати/відкривати дані, якими користувач хоче поділитися;
- сповіщення користувачів при зміні важливих даних;
- можливість відправлення текстового повідомлення користувачам;

- інтерфейс застосунку повинен бути розроблений англійською та українською мовами.

2.8 Висновки до розділу

На етапі вибору та обґрунтування системи, що проєктується, було зроблено такі висновки:

- мовами програмування вебзастосунку була обрано TypeScript та C#;
- клієнтська частина проєкту буде розроблена за допомогою фреймворку Nuxt.js, використовуючи мову програмування TypeScript;
- серверна частина проєкту буде розроблена за допомогою фреймворку asp .net core 6;
- у якості архітектури застосунку було обрано REST API;
- у якості системи зберігання даних було обрано MsSql, для швидкої та зручної роботи з базою даних був обраний фреймворк Entity Framework;
- у якості середовища розробки, буде використовуватись Visual studio code для розробки клієнтської частини та Visual studio для розробки серверної частини.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Розділ 3 описує алгоритми функцій програми та обґрунтування прийнятих в процесі розробки рішень щодо реалізації компонентів системи, визначених на етапі проєктування.

3.1 Алгоритм функціонування програми

Порядок прийому та обробки запитів до серверу регламентується стандартом WSGI (з англ. «Web Server Gateway Interface»), розробленого з метою опису взаємодії вебзастосунків, написаних на мові програмування Javascript. Схема роботи WSGI-застосунків зображена на рисунку 3.2.

Як було визначено на етапі проєктування, застосунок складається з двох незалежних модулів: клієнтської частини та серверної частини. Обмін даними та взаємодія між цими компонентами здійснюється за допомогою протоколу передачі даних HTTP, який побудований таким чином, що клієнт надсилає запит, а сервер відповідає на цей запит.

Схематичне зображення взаємодії компонентів системи зображена на рисунку 3.1.

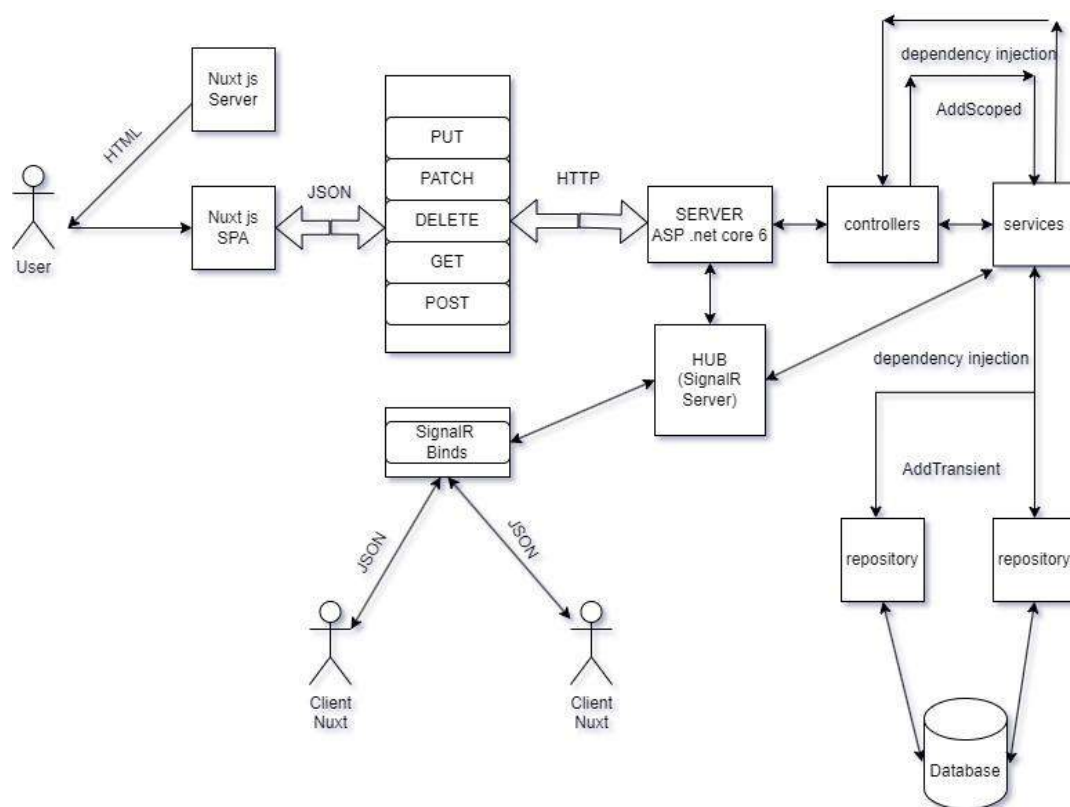


Рисунок 3.1 – Взаємодія компонентів системи

Надсилаючи запит, клієнт спочатку отримує HTML-сторінку, згенеровану клієнтським застосунком (а також файли стилів .css і коду .js, пов'язані з цією сторінкою). Потім, використовуючи Axios (бібліотека обробки HTTP-запитів), клієнт надсилає необхідні дані (дані, що зберігаються в базі даних і надсилаються на сервер для обробки, подальшого зберігання тощо, або інформацію, необхідну для заповнення поточної сторінки) і надсилає запит на сервер для отримання або обробки даних.

Цей підхід широко використовується при створенні клієнт-серверних застосунків.

Потрапивши до asp .net core серверу запит спочатку обробляється проміжними шаром (англ. «Middleware»), що видозмінює запити перед їх потраплянням до відповідного контролера, після цього контролер передає дані до сервісу, в сервісі дані оброблюються, після цього сервіс відправляє дані до репозиторія, в репозиторії за допомогою вхідних даних та Entity Framework, створюється запит до бази даних, після отримання даних від бази, репозиторій

передає дані назад до сервісу, в сервісі дані з бази оброблюються, після цього сервіс або ж передає дані до контролеру, де контролер дає відповідь клієнту, або ж визиває новий репозиторій, після визовів репозиторіїв та отримання усіх даних, оброблені дані все рівно потрапляють назад до контролеру. Необхідний контролер обирається в залежності від того, до якої адреси був направлений запит.

Контролер, використовуючи інтерфейс моделі для роботи з даними і реалізації бізнес-логіки, тобто правил обробки інформації, відправляє відповідь клієнту, в текстовому форматі JSON з вказанням коду статусу HTTP (наприклад, 200 у випадку вдалого запиту, 400 – у випадку некоректно наданих даних, 403 через заборону звернення і так далі). Алгоритм взаємодії окремих модулів серверної частини (представлення, контролерів, моделі) зображено на рисунку 2.2.

Також на сервері запускається вебсокет сервер SignalR, за його допомогою можна миттєво відправити запити до всіх користувачів в так званих групах.

SignalR - це безкоштовна програмна бібліотека з відкритим вихідним кодом для Microsoft ASP.NET, яка дозволяє надсилати асинхронні сповіщення з серверного коду до клієнтських вебзастосунків. Бібліотека включає серверні та клієнтські JavaScript компоненти.

Для приведення отриманих даних до Javascript-об'єктів, а також структурування відповідей до формату JSON у NodeJs використовуються серіалізатори (англ. «serializers»). Вони також відповідають за початкову валідацію даних, їх відповідність вимогам [5].

3.2 Основні рішення щодо реалізації компонентів системи

3.2.1 Основні рішення щодо реалізації серверної частини

Asp .net Core надає можливість розбиття застосунку на окремі структурні частини та реалізує у собі архітектуру Rest. Для зручної роботи з застосунком .net

core надає можливість розбивання коду на так звані слої (layers). Нижче наведено перелік створених слоїв:

- BLL (Business logic layer) – система класів, в якій описана бізнес логіка застосунку;
- CloudinaryAPI – система класів, в якій описана логіка роботи з хмарним сховищем Cloudinary;
- CORE – система класів, в якій описані усі константи, перелічувані типи, розширення та хелпери для застосунку;
- DAL (Data access layer) – система класів, в якій описані моделі бази даних, міграції та основний файл для роботи з базою даних, в якому описані зв'язки таблиць, а також властивості полів;
- Diploma server – основний шар застосунку, в якому знаходиться файл конфігурації застосунку, контролери, вебсокет сервер, файли інфраструктури, та моделі, які повертаються клієнту у вигляді JSON;
- Diploma server Test – шар, у якому знаходяться юніт тести застосунку;
- Repositories – система класів, в якій описана логіка роботи з базою даних.

Для прийняття запитів від клієнта, на стороні серверу використовується так звані контролери. Відстежити список контролерів можна за допомогою відкриття контролер файлу, або ж за допомогою спеціально бібліотеки Swagger, ця бібліотека підключається до asp .net core з коробки. Бібліотека Swagger генерую статичну сторінку під час запуску застосунку у режимі development, Swagger сторінка застосунку зображена на рисунку 3.2.

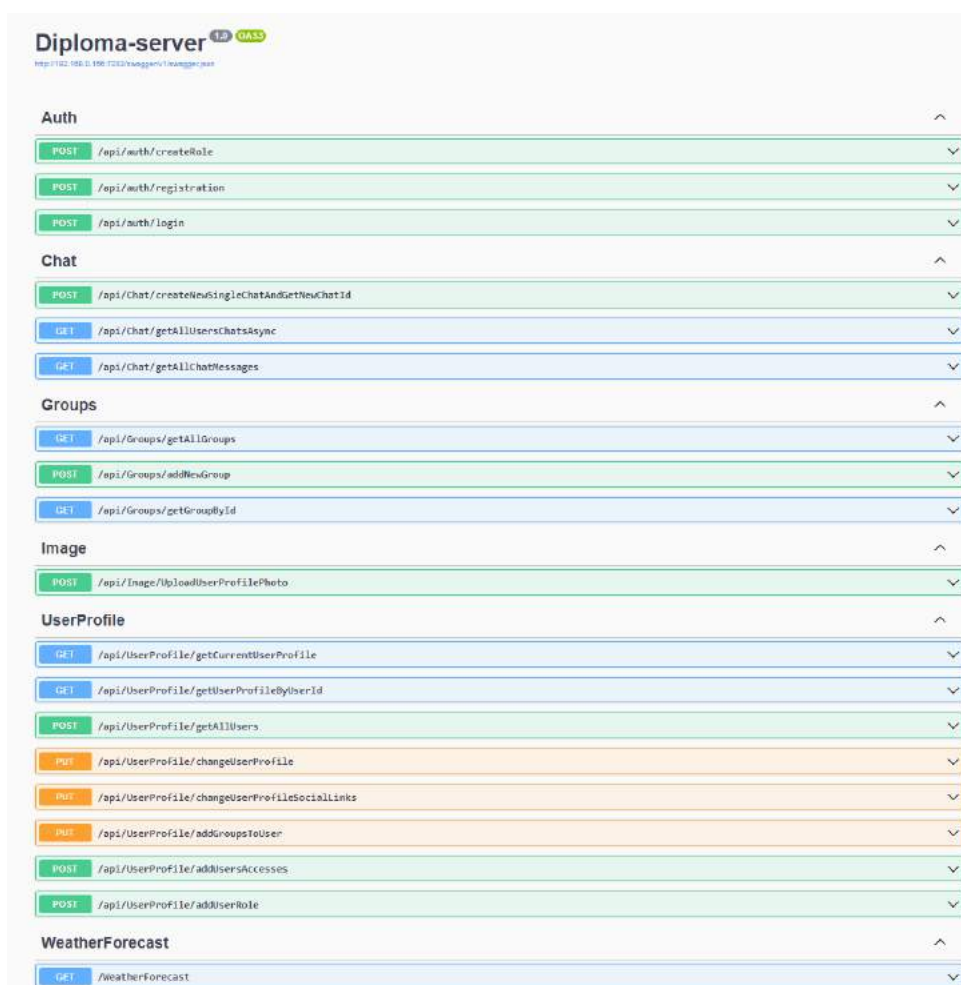


Рисунок 3.2 – Сторінка Swagger застосунку

Для обробки запитів на виконання CRUD операцій до серверу, реалізована наступна модель заявок за відносним посиланням /controllers/:

- при відправлені POST запиту за посиланням /api/auth/createRole буде добавлена нова роль для користувачів застосунку;
- при відправлені POST запиту за посиланням /api/auth/login з авторизаційною моделлю користувача, в яку входить електронна пошта і пароль, система спочатку перевірить наявність логіна у базі, якщо ж такий логін буде знайдений у базі, система захешує отриманий з клієнтської частини пароль та зрівнює хеш паролю наявний у базі даних, та при успішному виконанні поверне клієнту JWT token;

- при відправленні POST запиту за посиланням `/api/auth/registration` з моделлю нового користувача, модель спочатку буде провалідирована а потім, якщо ж модель буде валідною, буде створений новий користувач системи;
- при відправленні GET запиту за посиланням `/api/Chat/createNewSingleChatAndGetNewChatId` система створює новий чат, та поверне ідентифікаційний номер нового чату клієнту;
- при відправленні GET запиту за посиланням `/api/Chat/getAllUsersChatsAsync` будуть отримані користувачі чату за його ідентифікаційним номером;
- при відправленні GET запиту за посиланням `/api/Chat/getAllChatMessages` будуть отримані всі повідомлення за чатом;
- при відправленні GET запиту за посиланням `/api/Groups/getAllGroups` будуть отримані всі групи, які є на порталі;
- при відправленні POST запиту за посиланням `/api/Groups/addNewGroup` буде додана нова група за її назвою;
- при відправленні GET запиту за посиланням `/api/Groups/getGroupById` буде отримана вся інформація по групі за її ідентифікаційним номером;
- при відправленні POST запиту за посиланням `/api/Image/UploadUserProfilePhoto` буде замінена старий аватар користувача на новий, за його посиланням на CloudinaryAPI;
- при відправленні GET запиту за посиланням `/api/UserProfile/getCurrentUserProfile` буде отриманий користувач за його JWT токеном;
- при відправленні GET запиту за посиланням `/api/UserProfile/getUserProfileByUserId` буде отриманий користувач за його ідентифікаційним номером;
- при відправленні PUT запиту за посиланням `/api/UserProfile/changeUserProfile` буде змінена основна інформація користувача за його новою моделлю;

- при відправленні GET запиту за посиланням `/api/UserProfile/getAllUsers` будуть отримані всі студенти за їх ролями;
- при відправленні PUT запиту за посиланням `/api/UserProfile/changeUserProfileSocialLinks` буде змінена інформація користувача про його соціальні мережі;
- при відправленні POST запиту за посиланням `/api/UserProfile/addGroupsToUser` користувач буде доданий до нової учнівської групи;
- при відправленні POST запиту за посиланням `/api/UserProfile/addUsersAccesses` будуть змінені доступи користувача, які він надає для публічного доступу;
- при відправленні POST запиту за посиланням `/api/UserProfile/addUserRole` користувачу буде присвоєна додаткова роль на порталі.

Для обробки запитів на виконання Web-sockets операцій до серверу, реалізована наступна модель зв'язку клієнта з сервером:

- при використанні `connection.invoke('Ping')`, клієнт відправляє строку Ping до вебсокет серверу, це потрібно для перевірки на те, що користувач ще онлайн;
- при використанні `connection.on('Pong')`, клієнт підписується на приймання даних від вебсокет серверу, це потрібно для прийняття онлайн користувачів, які є на порталі;
 - при використанні `connection.invoke('SendNewMessage')` клієнт відправляє нове повідомлення до чату;
 - при використанні `connection.on('GetNewMessage')` клієнт підписується на прийняття нового повідомлення у чаті або ж на прийняття нотифікації нового повідомлення.

3.2.2 Основні рішення щодо реалізації клієнтської частини

Структура файлів і каталогів типової програми Nuxt виглядає наступним чином:

- «package.json» – список встановлених залежностей та режимів виконання програми;
- «node_modules/» – каталог для зберігання файлів залежностей (наприклад, сторонніх бібліотек, фреймворків тощо);
- «.nuxt/» – зібраний (скомпільований) проєкт у готовому вигляді, файли, що змінюються під час зборки знаходяться у директорії .nuxt/;
- «middleware/» – директорія з усіма проміжними шарами проєкту;
- «components/» – загальні компоненти системи;
- «store/» – директорій з сховищем станів проєкту;
- «models/» – директорія з усіма моделями проєкту;
- «core/enums/» – директорія з усіма константами у проєкті;
- «locales/» - директорія з усіма локалізаціями проєкту;
- «plugins/» - директорія з усіма плагінами системи;
- «pages/» - директорія з усіма сторінками системи;
- «services/» - директорія з сервісами проєкту;
- «layouts/» - директорія з усіма шаблонами проєкту;
- «core/const/» - директорія з константами проєкту;
- «components/» - директорія з усіма компонентами проєкту.

Для верстки (дизайну) вебзастосунка було використано графічний фреймворк Vuetify js. Фреймворк містить ряд готових компонентів (наприклад, спливаючі повідомлення, діалогові вікна, готові компоненти форм тощо) і надає можливість створити 12-колонкову розмітку вебзастосунку та адаптивний інтерфейс, в залежності від вимог поставленого завдання.

Для роботи з певними запитами з клієнтської частини на серверну, було використано `axios`. Модуль дозволяє клієнтській стороні надсилати запити (такі як `POST`, `GET`, `DELETE`, `PUT`, `PATCH`), а також приймати чи віддавати данні.

3.2.3 Основні рішення щодо проєктування бази даних

Важливою частиною проєктування будь-якої системи є рішення про організацію даних в ній. Для системи, що розробляється, було створено ER-діаграму (англ. “Entity Relation”), яка зображена на рисунку 3.3.

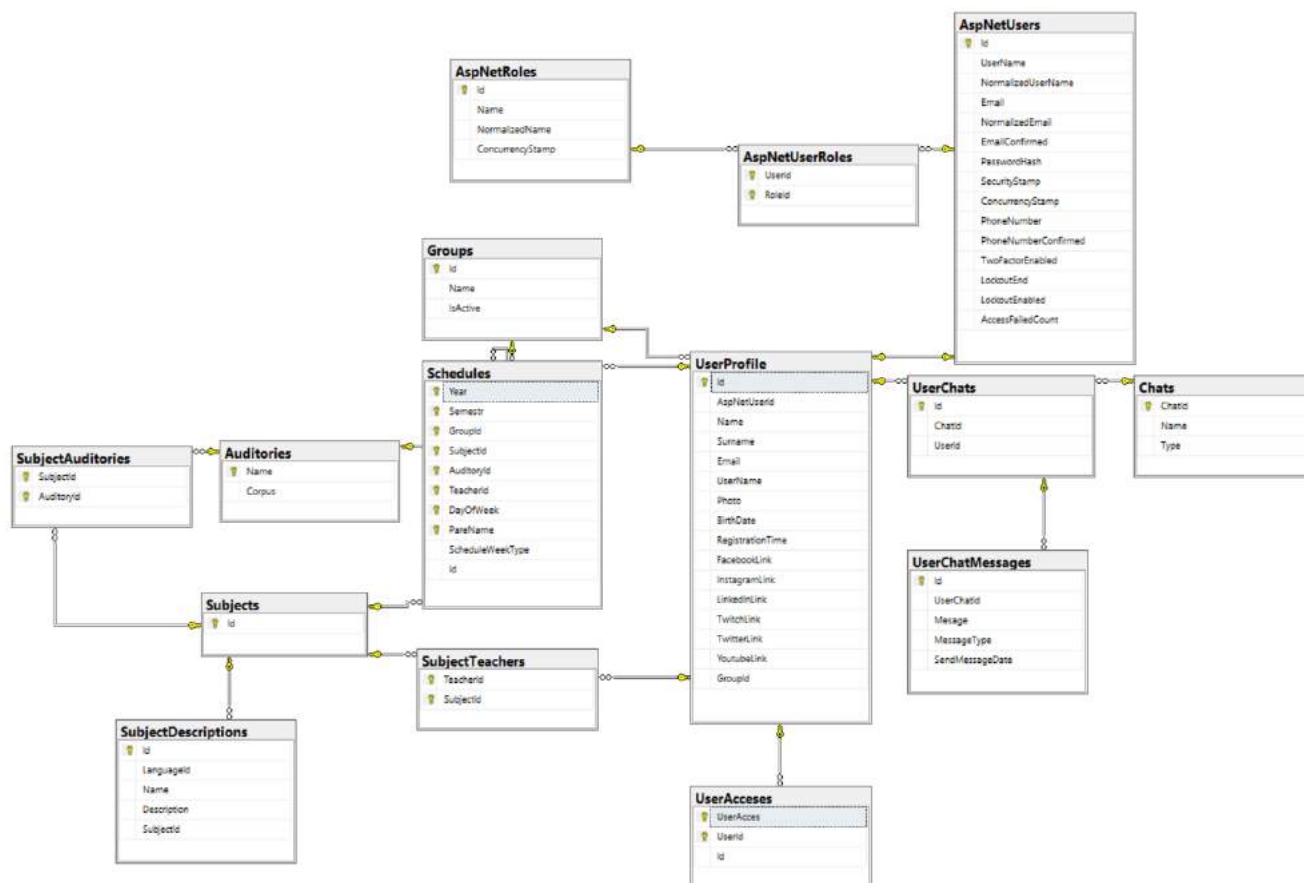


Рисунок 3.3 – ER-діаграма сутностей вебзастосунку

ER-діаграма описує концептуальну схему зв'язків між виділеними сутностями предметної області. Вона дозволяє не прив'язуючись до конкретної реалізації бази даних чи її системи керування (СКБД).

Нижче наведений перелік та опис створених сутностей:

- «AspNetRoles» – таблиця, в якій зберігаються ролі застосунку;
- «AspNetUserRoles» – таблиця, в якій зберігаються ролі користувачів;
- «AspNetUsers» – таблиця, в якій зберігаються користувачі застосунку;
- «UserProfile» – таблиця, в якій зберігаються основні дані користувачів;
- «UserChats» – таблиця, в якій зберігаються чати користувачів;
- «Chats» – таблиця, в якій зберігаються чати застосунку;
- «UserChatMessages» – таблиця, в якій зберігаються повідомлення користувачів;
- «UserAcces» – таблиця, в якій зберігаються доступи, які надав користувач;
- «Groups» – таблиця, в якій зберігаються групи застосунку;
- «Schedules» – таблиця, в якій зберігаються розклади застосунку;
- «Auditories» – таблиця, в якій зберігаються аудиторії застосунку;
- «SubjectAuditories» – таблиця, в якій зберігаються аудиторії предметів;
- «Subjects» – таблиця, в якій зберігаються предмети застосунку;
- «SubjectDescriptions» – таблиця, в якій зберігаються опис предметів;
- «SubjectTeachers» – таблиця, в якій зберігаються предмети викладачів.

3.2.4 Основні рішення щодо створення інтелектуального чату

Інтелектуальний чат - це чат, який відповідає на команди та ключові слова, це простий інтелектуальний інтерфейс, призначений для взаємодії з користувачами на основі конкретних питань та пропозицій. Його основна функція – розпізнавати введені користувачем дані та реагувати на них відповідно до певних шаблонів або конкретних команд.

Інтелектуальний чат вебзастосунку може обробляти такі команди:

- «Яка зараз пара» - видає студенту, яка зараз пара у його групі;

- «Які викладачі зараз викладають» - видає студенту перелік викладачів, які зараз викладають;
- «Які аудиторії у N корпусі вільні на X пару у А-день» - видає користувачу перелік аудиторій, які вільні у N корпусі на X парі за А день тижня;
- «Відключи оповіщення чату» - вимикає оповіщення чату;
- «Додай нову групу з назвою (назва групи)» - додає нову групу за назвою групи;
- «Додай нову аудиторію з назвою (назва аудиторії) та корпусом N» - додає нову аудиторію за назвою та корпусом N;
- «Додай нового викладача (ідентифікатором користувача)» - додає роль викладача до користувача за ідентифікатором користувача;
- «Зроби активною (назва налаштування приватності)» - включає налаштування приватності за назвою;
- «Вимкни (назва налаштування приватності)» - вимикає налаштування приватності за назвою;
- «Додай (назва соціальної мережі) за (посиланням)» - додає посилання на соціальну мережу за її назвою та посиланням.

3.3 Підсумки опису програми

На етапі реалізації програми були розроблені попередньо визначені програмні компоненти. Для спрощення процесу розробки були використані готові рішення, доступні у вигляді модулів для використання ядра .net та Nuxt, як описано в розділі 3.2.

На момент завершення етапу впровадження програмне забезпечення відповідало всім функціональним і нефункціональним вимогам, викладеним в технічному завданні. Розроблений програмний продукт відповідає сучасним тенденціям у розробці вебзастосунків і може бути легко модифікований у майбутньому.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

4.1.1 Призначення та функції

Вебзастосунок забезпечує виконання таких функцій:

- функція авторизації;
- функція реєстрації;
- функція надавання доступів до різних даних;
- функція створення чатів;
- функція оповіщення;
- функція взаємодії клієнтів один з одним за допомогою вебсокет серверу;
- функція швидкого розвертання вебзастосунку на хост-сервері;
- функція перегляду відкритої інформації;
- функція сповіщення програміста про виникнення помилок на серверній стороні;
- функція сповіщення за допомогою вебсокет серверу.

4.1.2 Умови виконання програми

Для коректної роботи програмного забезпечення необхідні наступні системні параметри та периферійні пристрої:

- одноядерний процесор з тактовою частотою 2.4 ГГц та більше;
- обсяг оперативної пам'яті не менше ніж 512Мб;
- жорсткий диск не менш ніж з 50 Гб пам'яті;
- блок живлення не менш ніж 240 ватт;
- Windows 7 та вище.

4.2 Характеристики програми

Система розрахована на цілодобовий режим роботи протягом 365 днів на рік. Засобами контролю моніторингу є автоматизовані вбудовані в платформу програмні компоненти, які самостійно виконують такі дії в разі помилки:

- якщо відсутнє підключення до бази даних, то система видає помилку адміністратору у консолі серверу про те, що підключення відсутнє;
- якщо виникли помилки при підключенні клієнта до вебсокет серверу система видає помилку у консолі серверу;
- якщо виникла якась системна помилка, то система видає помилку адміністратору у консолі серверу про те, яка сама помилка виникла.

Для відновлення програми потрібно перезапустити сервер та запустити програму через консоль за допомогою програми docker, використовуючи команди:

- «docker build .» - створює докер image;
- «docker images» - показує усі створенні контейнери docker;
- «docker run -d -p 3000:3000 --name diploma --rm назва створеного контейнеру» - розвертає докер контейнер за створеним imageм.

Обсяг пам'яті, якій займає програма – 270 Мб.

Програма написана на мовах програмування C# та TypeScript, за допомогою серверу .net core 6 та NodeJs.

Перелік шарів серверної частини, з яких складається програма:

- BLL (Business logic layer) – система класів, в якій описана бізнес логіка застосунку;
- CloudinaryAPI – система класів, в якій описана логіка роботи з хмарним сховищем Cloudinary;
- CORE – система класів, в якій описані усі константи, перелічувані типи, розширення та хелпери для застосунку;

- DAL (Data access layer) – система класів, в якій описані моделі бази даних, міграції та основний файл для роботи з базою даних, в якому описані зв'язки таблиць, а також властивості полів;

- Diploma server – основний шар застосунку, в якому знаходиться файл конфігурації застосунку, контролери, вебсокет сервер, файли інфраструктури, та моделі, які повертаються клієнту у вигляді JSON;

- Diploma server Test – шар, у якому знаходяться юніт тести застосунку;

- Repositories – система класів, в якій описана логіка роботи з базою даних.

Перелік файлів клієнтської частини, з яких складається програма:

- package.json – список встановлених залежностей та режимів виконання програми;

- «node_modules/» – каталог для зберігання файлів залежностей (наприклад, сторонніх бібліотек, фреймворків тощо);

- «.nuxt/» – зібраний (скомпільований) проєкт у готовому вигляді, файли, що змінюються під час зборки знаходяться у директорії .nuxt/;

- «middleware/» – директорія з усіма проміжними шарами проєкту;

- «components/» – загальні компоненти системи;

- «store/» – директорій з сховищем станів проєкту;

- «models/» – директорія з усіма моделями проєкту;

- «core/enums/» – директорія з усіма константами у проєкті;

- «locales/» - директорія з усіма локалізаціями проєкту;

- «plugins/» - директорія з усіма плагінами системи;

- «pages/» - директорія з усіма сторінками системи;

- «services/» - директорія з сервісами проєкту;

- «layouts/» - директорія з усіма шаблонами проєкту;

- «core/const/» - директорія з константами проєкту;

- «components/» - директорія з усіма компонентами проєкту.

4.3 Звернення до програми

Управління вмістом системи відбувається лише через завантажену програму. Для того, щоб передати повне управління над програмою потрібно лише запустити `docker` команди, які вказані у розділі 4.2, за допомогою файлу `DOCKERFILE`, усі скрипти запускаються за замовчуванням, через настроєний файл конфігурації `docker`, який вже знаходиться у серверній та клієнтській частині проекту. Для успішного запуску програми, `docker` команди, потрібно виконати у кореневій папці клієнтської та серверної частини проекту.

4.4 Вхідні та вихідні дані

Вхідними даними програми є основні дані про користувача, соціальні мережи користувача, доступи, які надає користувач до своїх даних, назви груп, назви предметів, дані про розклад, ролі користувача, аудиторії ЗВО та повідомлення користувачів.

Вихідними даними програми є списки користувачів, розкладів, викладачів, відкритих даних, аудиторій, та повідомлень.

4.5 Повідомлення програмісту

Усі повідомлення, що може отримати програміст представлений нижче:

- «Listening on: `http://172.17.0.2:3000/`» – означає, що програма запущена по ір адресі 172.17.0.2 з портом 3000 на протоколі `http` (рис. 4.1);
- «Information: `WebSocket connected to ws://95.47.123.70:7283/`» – означає, що клієнт підключився до вебсокет серверу (рис. 4.2).
- «An unexpected error occurred invoking 'Ping' on the server» – означає, що на стороні вебсокет серверу виник збій (рис. 4.3).

– «An exception occurred while iterating over the results of a query for context type 'DAL.ApplicationContext'» – означає що виникла помилка при підключенні до бази даних (рис 4.4).

Приклад системних помилок показано на рисунку 4.1 – 4.3.

```

Logs    Inspect    Bind mounts    Exec    Files    Stats
2023-11-28 03:20:46
2023-11-28 03:20:46 > Diploma@1.0.0 start
2023-11-28 03:20:46 > nuxt start
2023-11-28 03:20:46
2023-11-28 03:20:48 i Listening on: http://172.17.0.2:3000/

```

Рисунок 4.1 – Приклад системних повідомлень під час успішного запуску програми

```

2023-11-28 03:23:54 i [2023-11-28T01:23:54.962Z] Information: WebSocket connected to ws://95.47.123.70:7283/chat?id=I3iJ1_YLjyAQ2fPJYpv9Rw.
2023-11-28 03:23:54 123
2023-11-28 03:24:01 i [2023-11-28T01:24:01.121Z] Information: WebSocket connected to ws://95.47.123.70:7283/chat?id=H3aRu7jB1bY8iVR245bX_w.

```

Рисунок 4.2 – Приклад системних помилок під час підключення до бази даних

```

2023-11-28 03:24:10
2023-11-28 03:24:10 ERROR An unexpected error occurred invoking 'Ping' on the server.
2023-11-28 03:24:10

```

Рисунок 4.3 – Приклад системних повідомлень під час успішного запуску програми

```

fail: Microsoft.EntityFrameworkCore.Database.Connection[20004]
An error occurred using the connection to database 'diploma' on server 'localhost'.

```

Рисунок 4.4 – Приклад системних повідомлень під час невдалого підключення до бази даних

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення та умови застосування програми

5.1.1 Призначення та функції

Дане ПЗ призначене для автоматизації та прискорення комунікації між студентами або викладачами ЗВО, а також для швидкого моніторингу змін у успішності, або для моніторингу змін у навчальному процесі.

Функції програмного забезпечення оператора, які розроблені у даному ПЗ:

- можливість авторизації користувачів;
- можливість реєстрації користувачів;
- можливість перегляду розкладу за студентами, групами, викладачами та аудиторіями;
- можливість створення чатів;
- можливість додавання предметів, аудиторій, викладачів, груп та розкладу;
- можливість перегляду створених предметів, аудиторій, розкладів, користувачів, груп, та додаткової інформації.
- можливість перегляду контенту на різних мовах;
- можливість створення розкладу по групам;
- можливість скривати/відкривати дані, якими користувач хоче поділитися;
- можливість сповіщення користувачів;
- можливість відправлення текстового повідомлення користувачам;
- можливість додавання мультимовного контенту.

5.1.2 Умови виконання програми

Для того, щоб виконання програмного засобу було коректне, необхідні такі параметри системи, а також такі периферійні пристрої як:

- однопоточний процесор з тактовою частотою 2.4 ГГц Та більше;

- обсяг оперативної пам'яті не менше ніж 512Мб;
- жорсткий диск не менш ніж з 50 Гб пам'яті;
- блок живлення не менш ніж 240 ватт;
- Windows 7 та вище.

5.2 Виконання програми

Для запуску програмного забезпечення необхідно лише встановити docker та запустити docker команди, які вказані у розділі 4.2, за допомогою файлу DOCKERFILE, усі скрипти запускатимуться за замовчуванням, через настроєний файл конфігурації docker, який вже знаходиться у серверній та клієнтській частині проєкту.

Для нормальної роботи програми у оператора повинно бути встановлений браузер з найновішою версією Javascript, Visual studio, .net Core та Docker.

Звертання до програми відбувається у той момент, коли користувач перейшов на сторінку вебзастосунку.

Для використання функцій програми оператор повинен перейти на головну сторінку вебзастосунку «<http://95.47.123.70:3000/>», у нього з'являється основна сторінка вебзастосунку (приклад наведено на рис. 5.1).



Рисунок 5.1 – Головна сторінка вебзастосунку

Для авторизації нового користувача у системі, потрібно спочатку зареєструвати нового користувача у системі, для цього треба спочатку перейти на сторінку входу (приклад наведено на рис. 5.2), а потім перейти на сторінку реєстрації (приклад наведено на рис. 5.3).

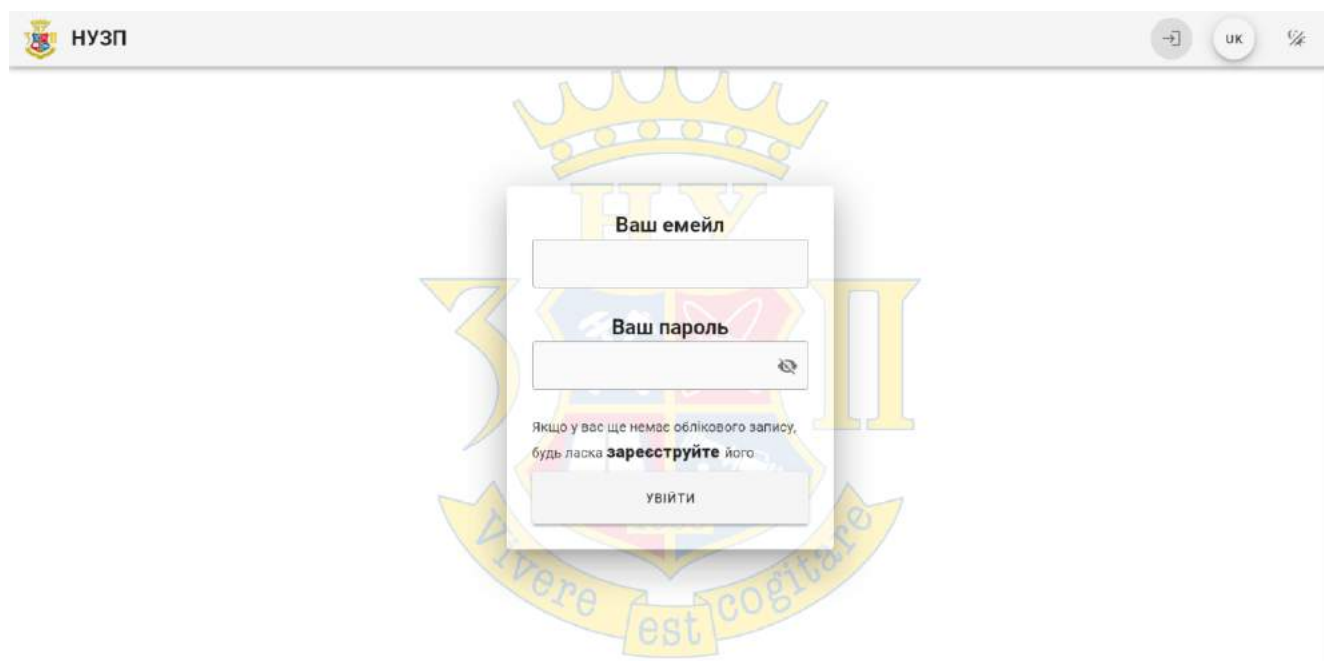


Рисунок 5.2 – Сторінка входу

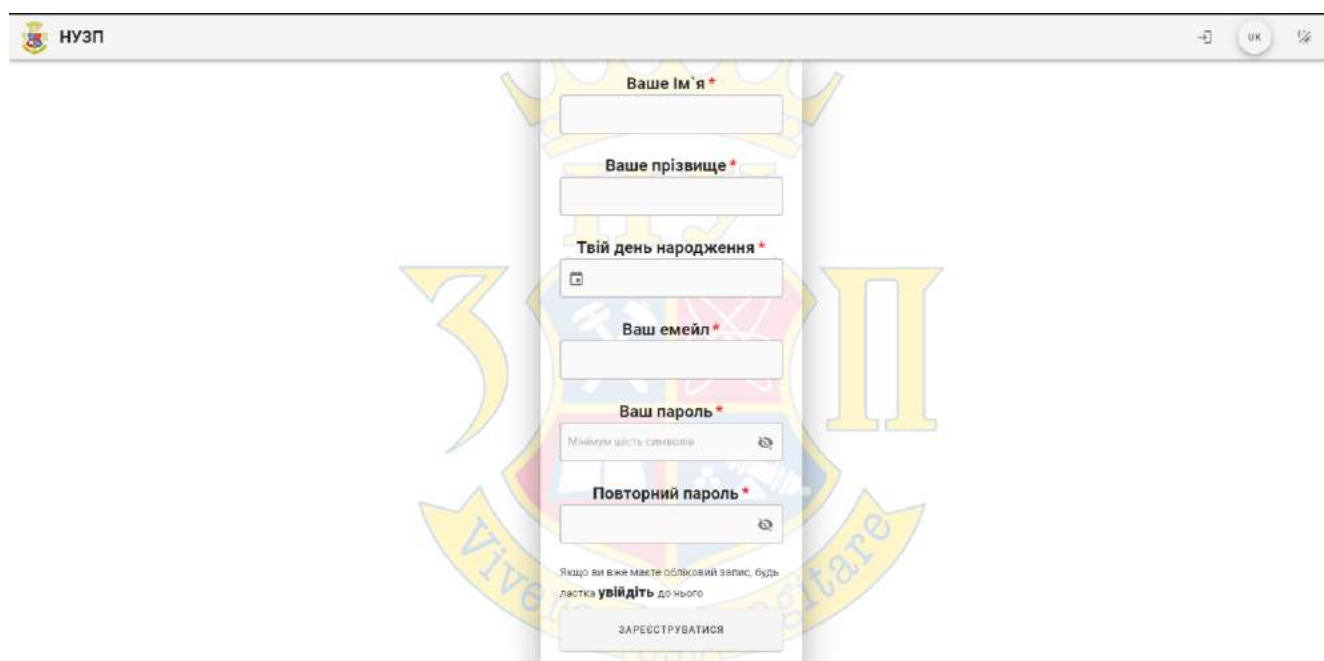


Рисунок 5.3 – Сторінка реєстрації

Для того, щоб змінити мову, треба натиснути на випадаючий список у шапці, після цього з списку мов обрати потрібну оператору мову (приклад наведено на рис. 5.4).

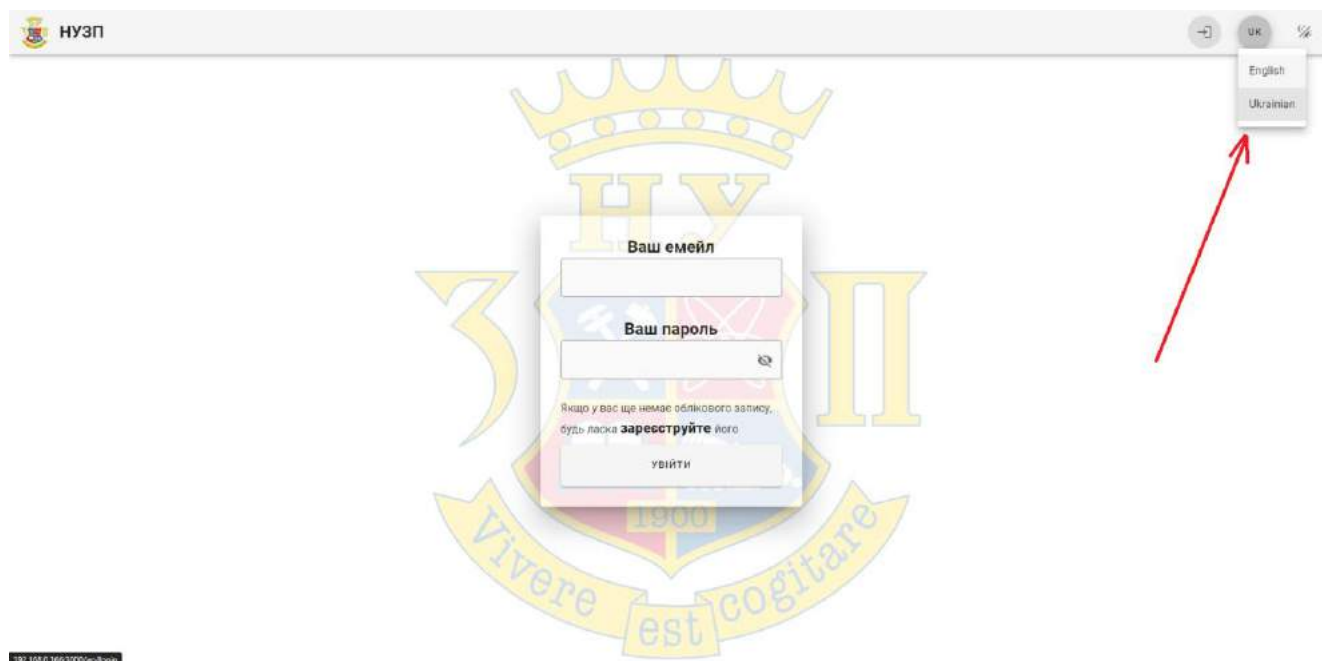


Рисунок 5.4 – Елемент зміни мови

Для того, щоб змінити тему, треба натиснути на спеціальну іконку у шапці, тема автоматично зміниться (приклад наведено на рис. 5.5).

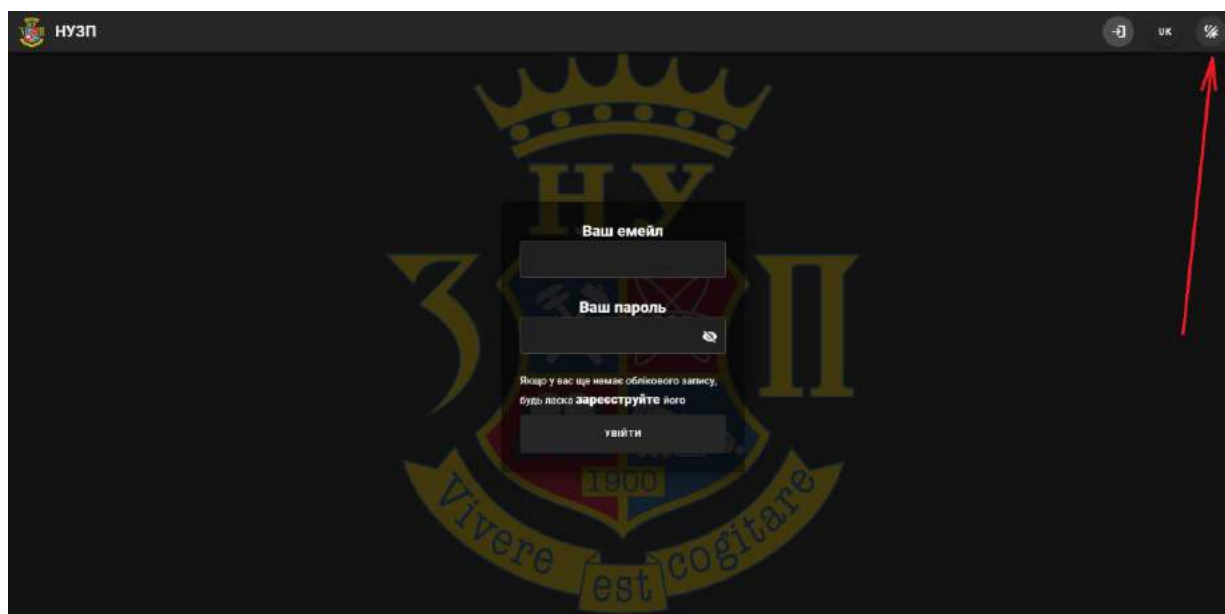


Рисунок 5.5 – Елемент зміни теми

Для перегляду інформаційної сторінки, оператор повинен перейти на вкладку Інфо, і після цього вибрати тему, якою цікавиться (приклад наведено на рис. 5.6).

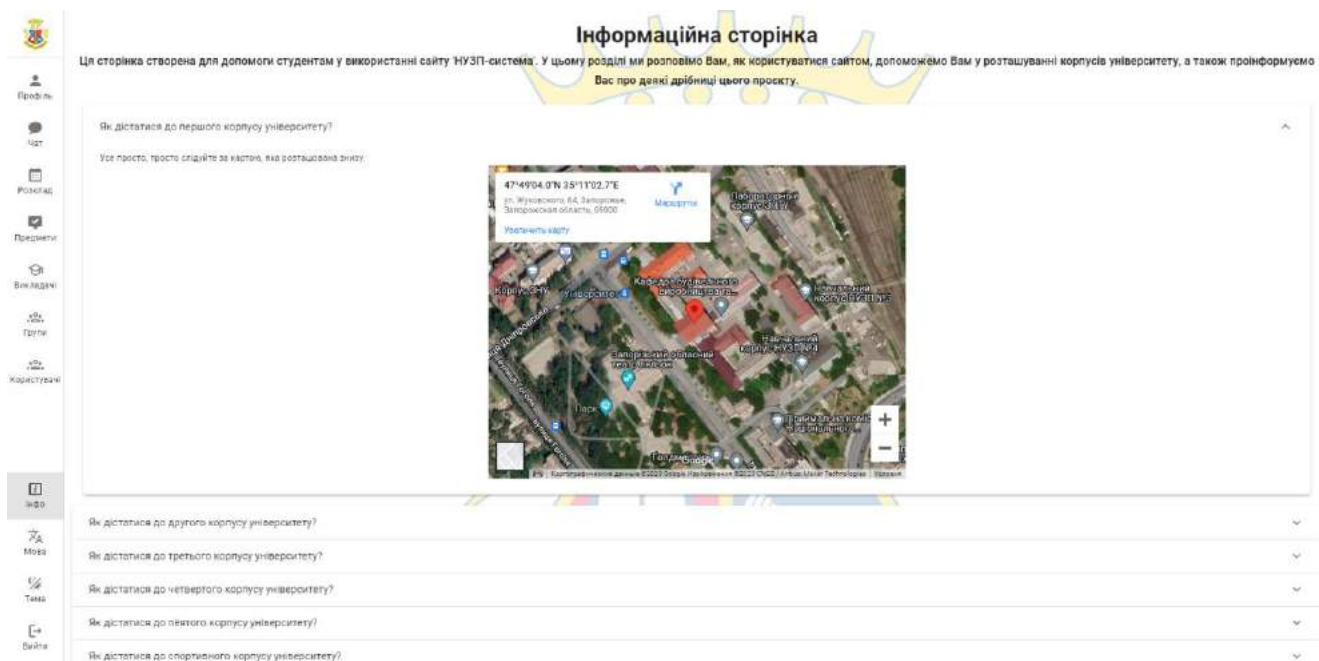
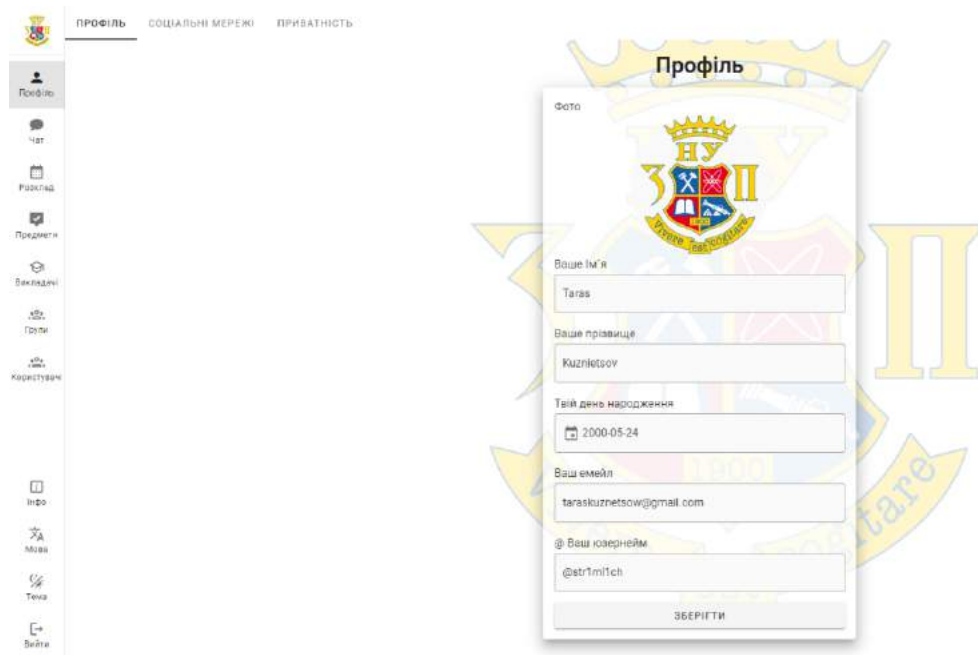


Рисунок 5.6 – Інформаційна сторінка

Для зміни чи перегляду основної інформації про свій аккаунт оператор повинен перейти на вкладку профіль, після цього він може змінити свої основні данні на підсторінці «Профіль» (приклад наведено на рис. 5.7). Після цього оператор може змінити дані про свої соціальні мережі, для цього оператор повинен перейти на підсторінку «Соціальні мережі» (приклад наведено на рис. 5.8). Для зміни налаштувань приватності, оператор повинен перейти на підсторінку «Приватність» (приклад наведено на рис. 5.9).



Профіль

ФОТО

Ваше ім'я
Taras

Ваше прізвище
Kuznetsov

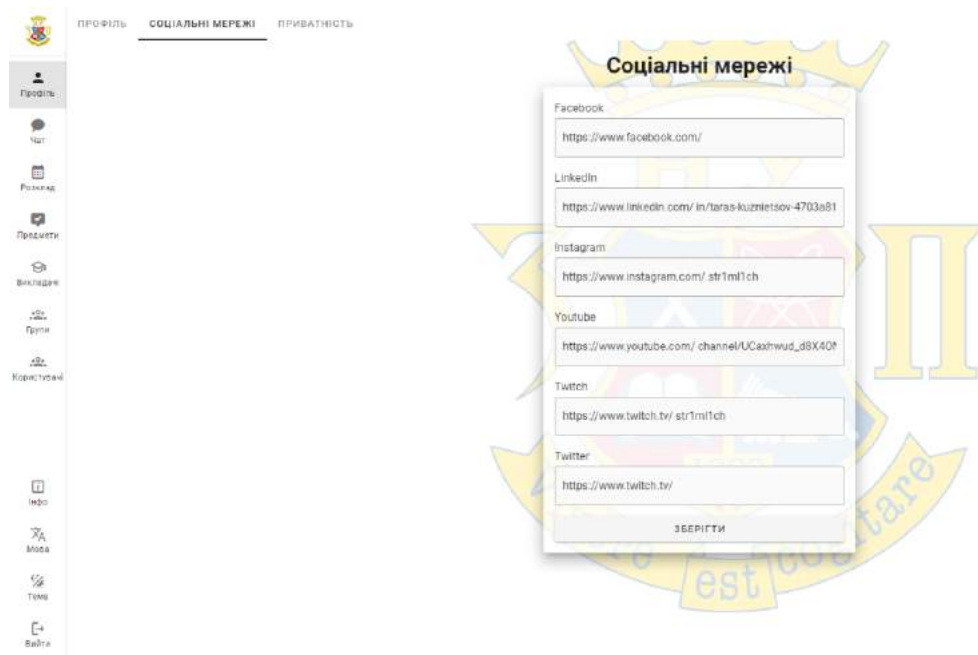
Ваш день народження
2000-05-24

Ваш емейл
taraskuznetsow@gmail.com

@ Ваш юзернейм
@str1ml1ch

ЗБЕРІГТИ

Рисунок 5.7 – Сторінка основної інформації оператора



Соціальні мережі

Facebook
<https://www.facebook.com/>

LinkedIn
<https://www.linkedin.com/in/taras-kuznetsov-4703a81>

Instagram
<https://www.instagram.com/str1ml1ch>

Youtube
https://www.youtube.com/channel/UCaxhwud_d8X40t

Twitch
<https://www.twitch.tv/str1ml1ch>

Twitter
<https://www.twitch.tv/>

ЗБЕРІГТИ

Рисунок 5.8 – Сторінка інформації соціальних мереж оператора

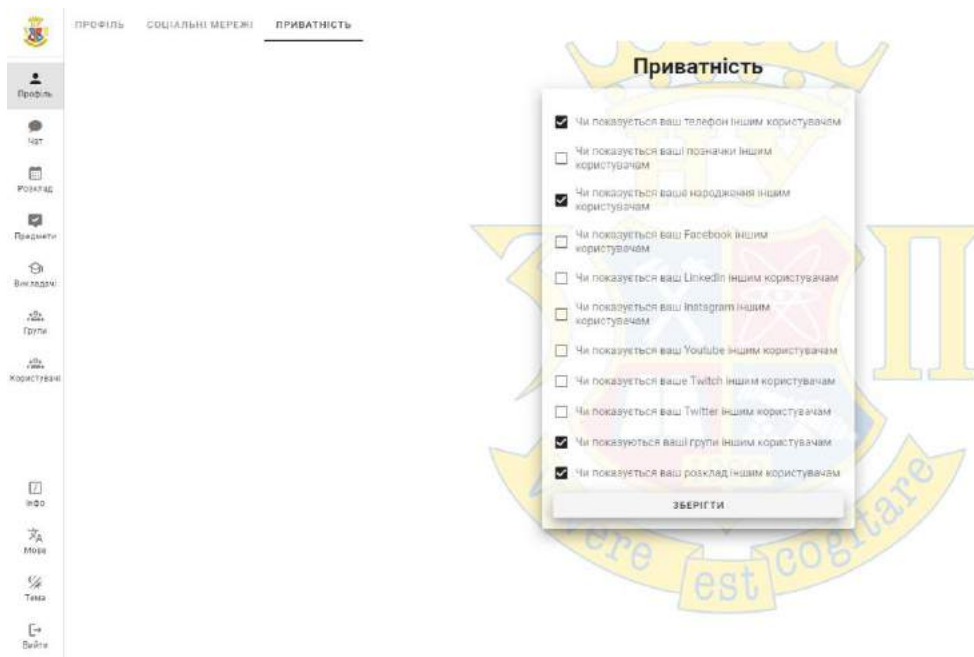


Рисунок 5.9 – Сторінка налаштування приватності оператора

Для того, щоб оператор міг подивитися список усіх користувачів вебзастосунку, оператор повинен перейти на вкладку «Користувачі», після цього обрати потрібного користувача, після цього оператор зможе побачити усі інформацію про іншого користувача, яку користувач надав для перегляду у налаштуваннях приватності (приклад наведено на рис. 5.10).

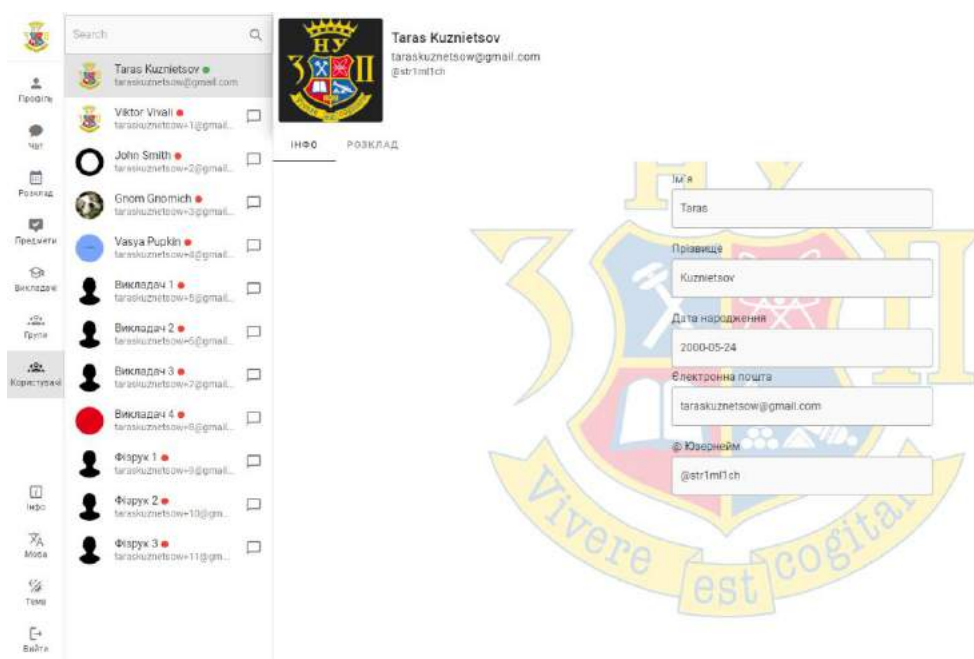


Рисунок 5.10 – Сторінка перегляду користувачів системи

Для перегляду інформації про студентські групи у системі, оператор повинен перейти на сторінку «Групи» (приклад наведено на рис. 5.11), для перегляду розкладу по групі, потрібно перейти на підсторінку «розклад» (приклад наведено на рис. 5.12).

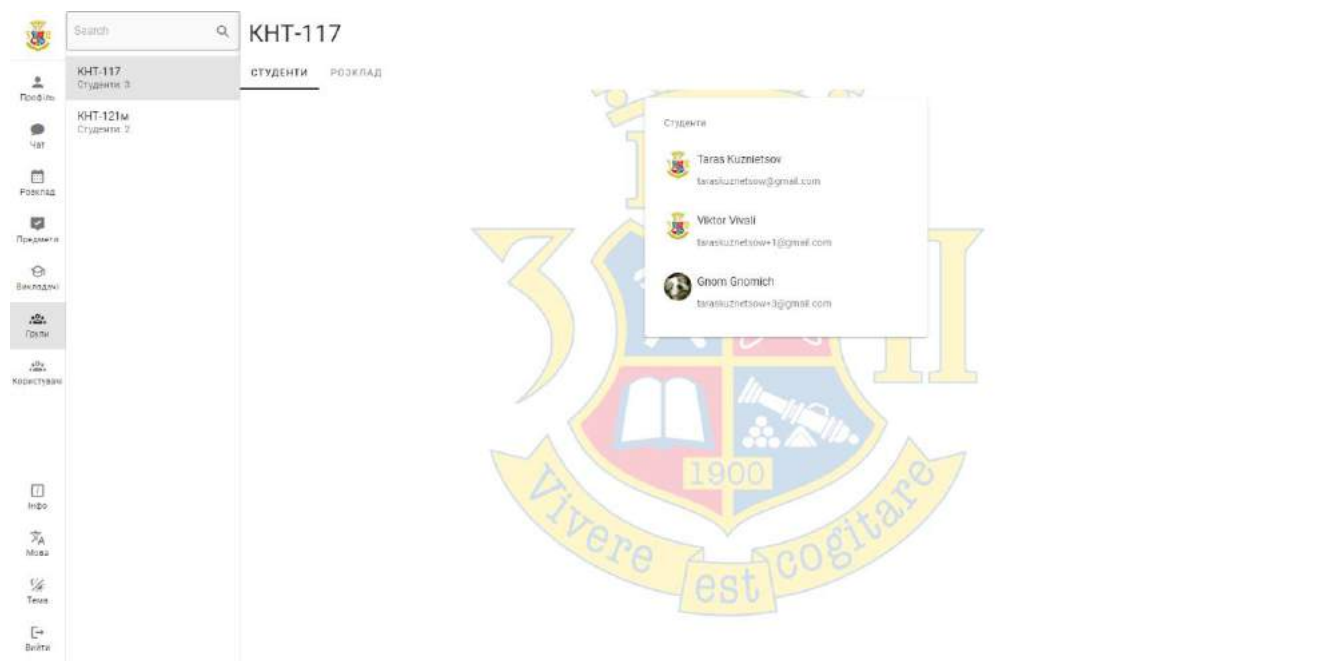


Рисунок 5.11 – Сторінка перегляду груп

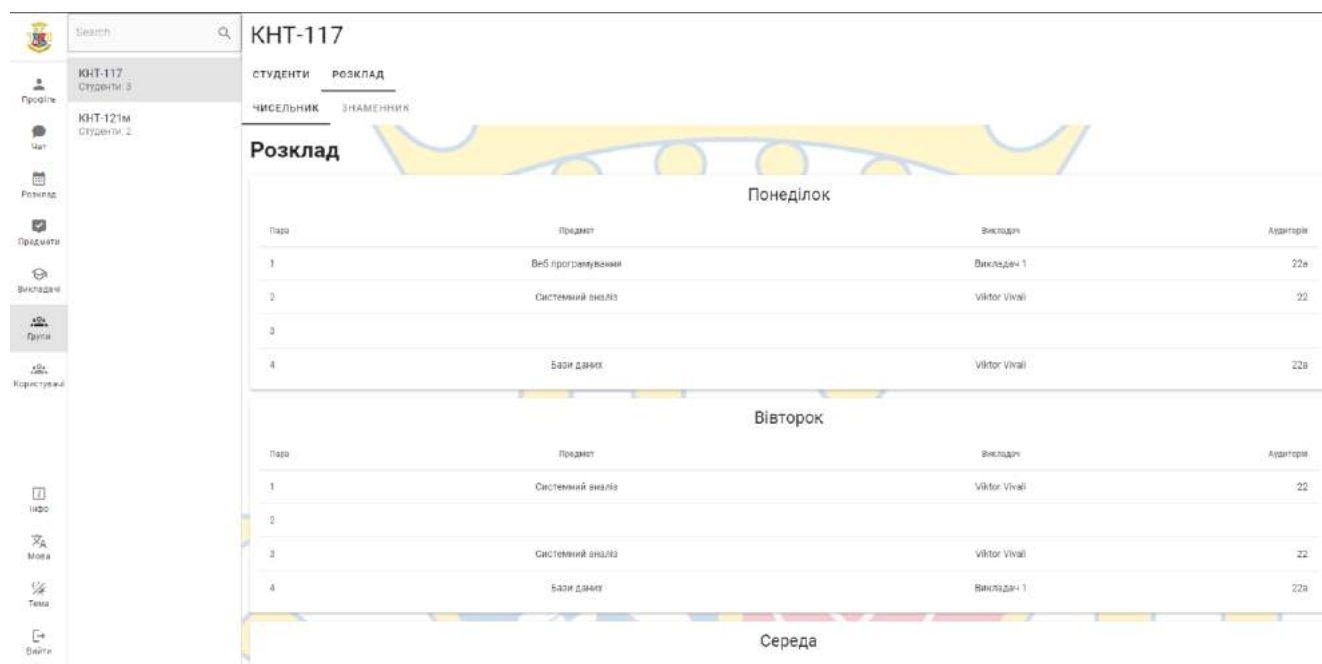


Рисунок 5.12 – Сторінка перегляду розкладу груп

Для перегляду інформації про викладачів у системі, оператор повинен перейти на сторінку «Викладачі» (приклад наведено на рис. 5.13), для перегляду розкладу за викладачем, потрібно перейти на підсторінку «розклад» (приклад наведено на рис. 5.14).



Рисунок 5.13 – Сторінка перегляду викладачів

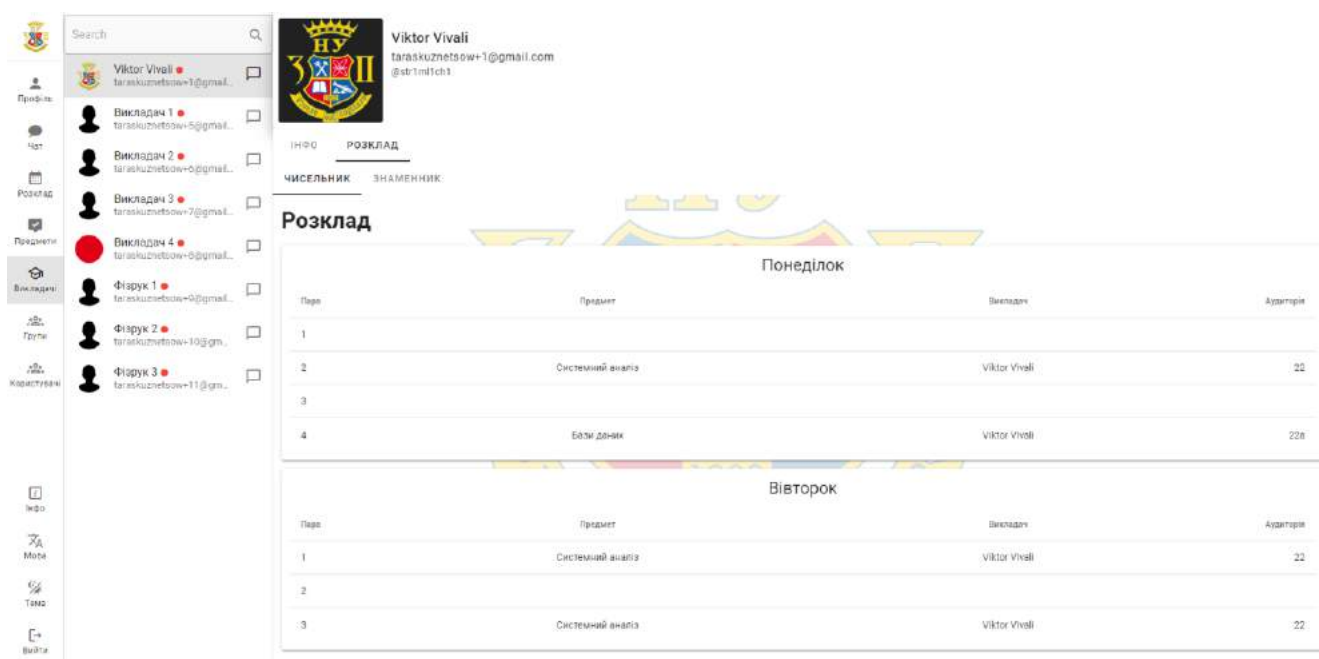


Рисунок 5.14 – Сторінка перегляду розкладу викладача

Для перегляду інформації про предмети, оператор повинен перейти на сторінку «Предмети» (приклад наведено на рис. 5.15).

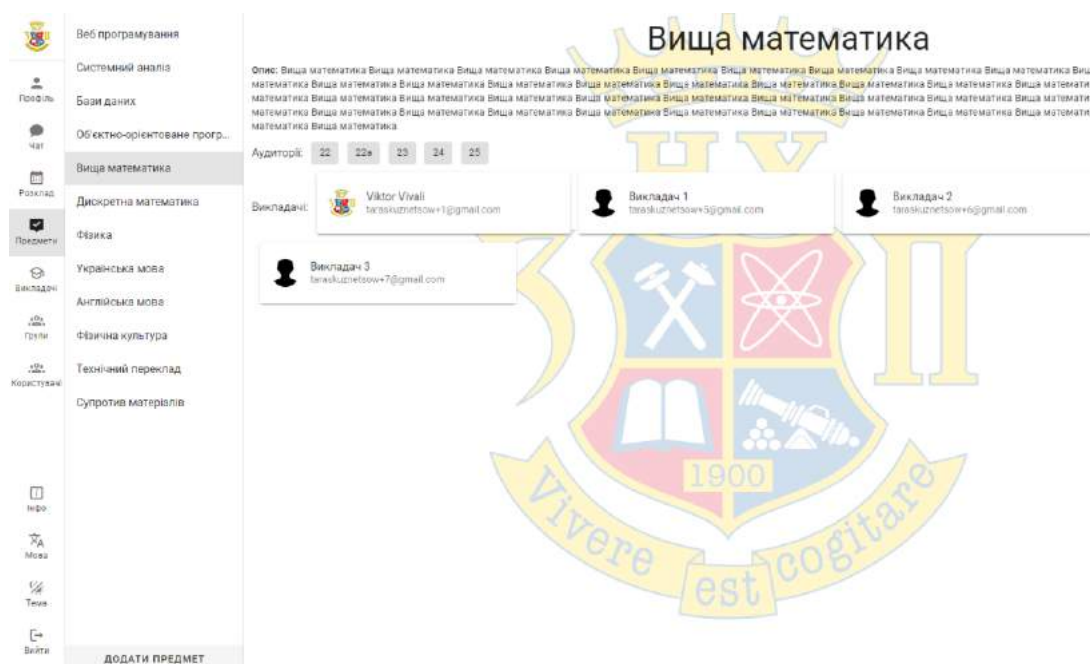


Рисунок 5.15 – Сторінка перегляду предметів

Для перегляду розкладу оператора, оператор має перейти на сторінку «Розклад» (приклад наведено на рис. 5.16).

ЧИСЕЛЬНИК

ЗНАМЕННИК

Профіль

Чат

Розклад

Примітки

Викладачі

Групи

Користувачі

INFO

Media

Tools

Вийти

Розклад

Понеділок

Пара	Предмет	Викладач	Аудиторія
1	Веб програмування	Викладач 1	22а
2	Системний аналіз	Viktor Vivali	22
3			
4	Бази даних	Viktor Vivali	22а

Вівторок

Пара	Предмет	Викладач	Аудиторія
1	Системний аналіз	Viktor Vivali	22
2			
3	Системний аналіз	Viktor Vivali	22
4	Бази даних	Викладач 1	22а

Середа

Пара	Предмет	Викладач	Аудиторія
1	Об'єктно-орієнтоване програмування	Викладач 1	22а
2			

Рисунок 5.16 – Сторінка перегляду розкладу оператора

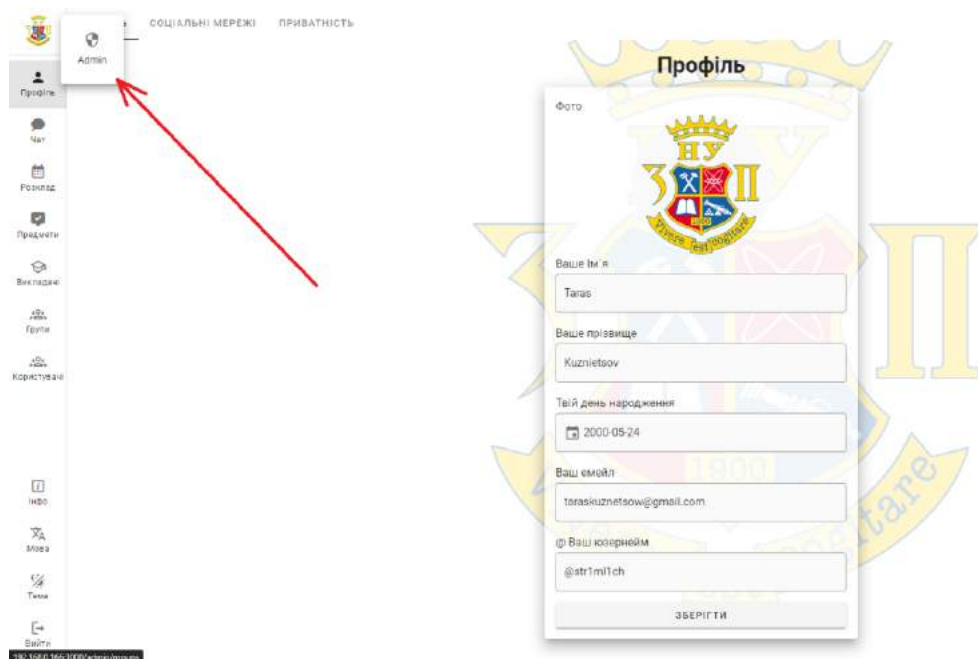


Рисунок 5.18 – Елемент переходу оператора до адмін ролі

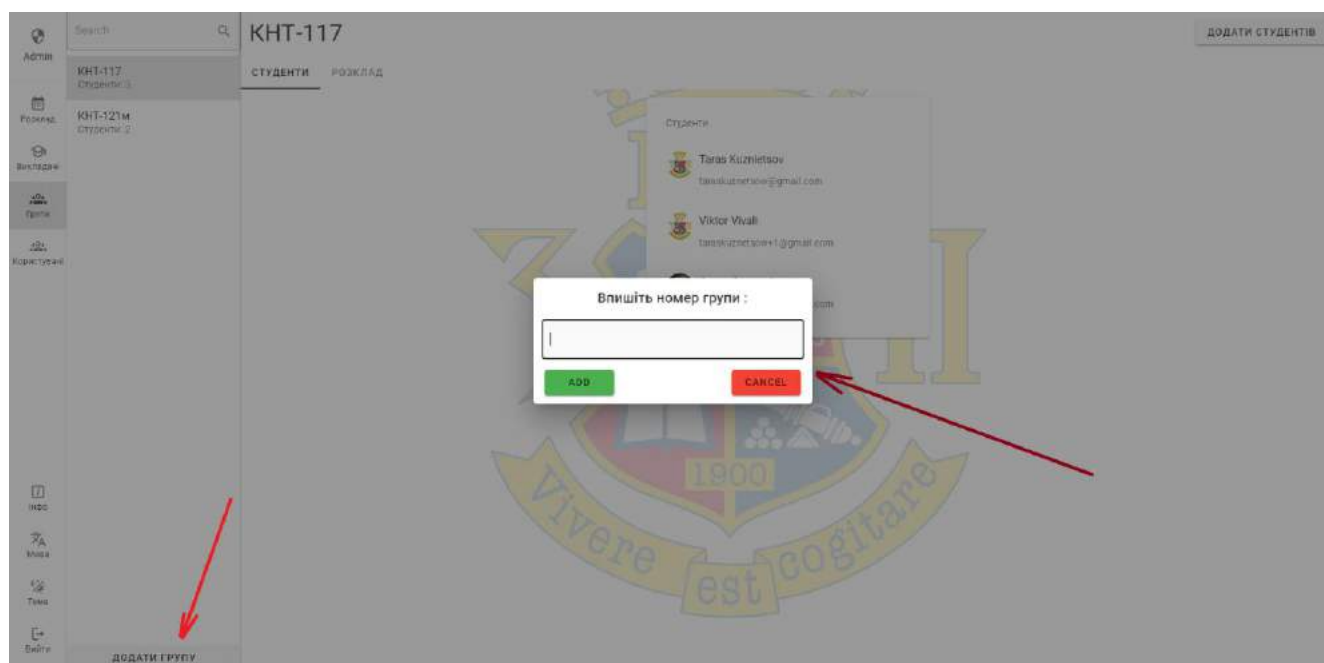


Рисунок 5.19 – Модальне вікно додавання нової групи

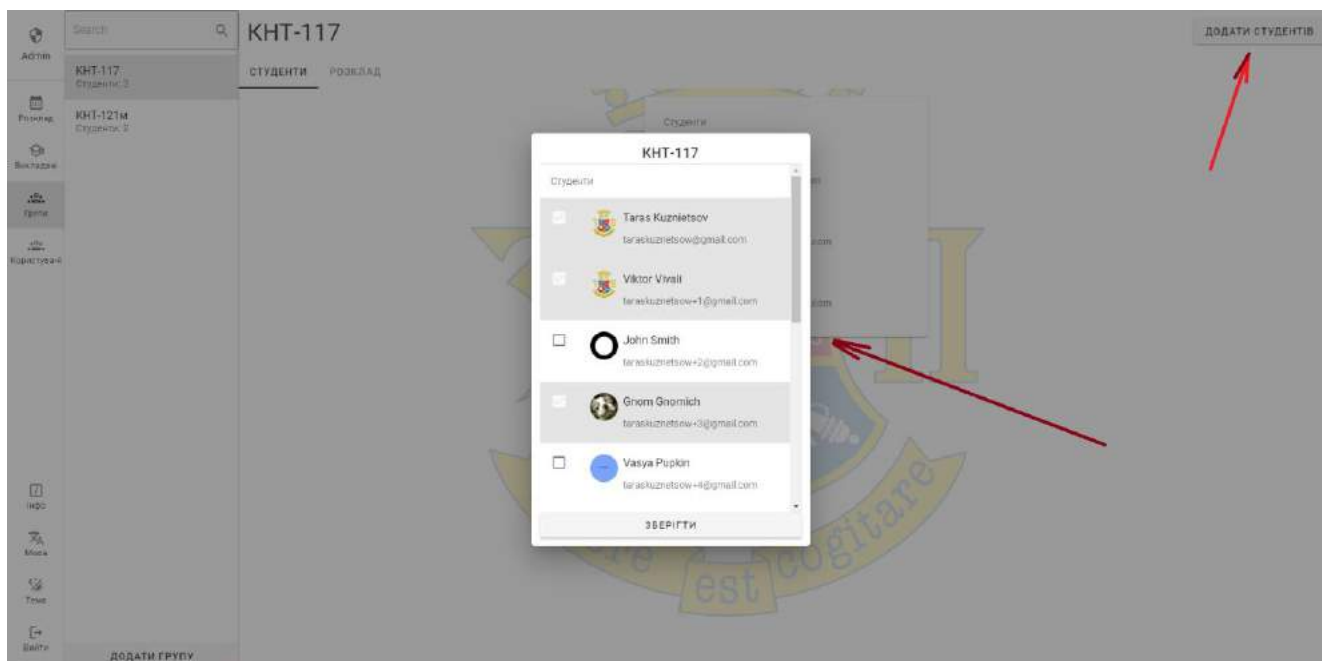


Рисунок 5.20 – Модальне вікно додавання студентів до групи

Для додавання нових викладачів, оператор має перейти на адмін-сторінку «Викладачі», після цього натиснути на кнопку «Додати викладача», після відкриття модального вікна, оператор має вибрати користувачів, яким треба додати роль викладача (приклад наведено на рис. 5.21).

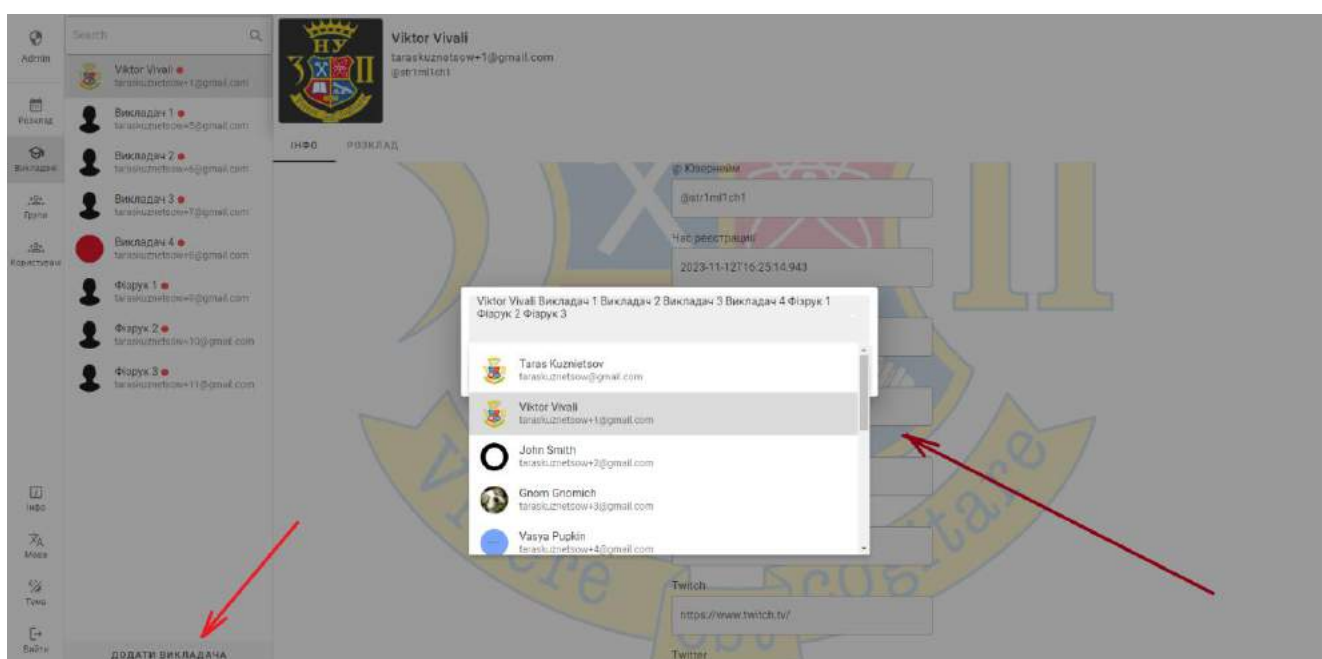


Рисунок 5.21 – Модальне вікно додавання викладачів

Для додавання нових предметів, оператор повинен перейти до сторінки «Розклад», після цього оператор повинен перейти до підсторінки «Предмети», після цього натиснути кнопку «Додати предмет», та у модальному вікні додати предмет (приклад наведено на рис. 5.22), для додавання аудиторій, оператор повинен перейти на підсторінку «Аудиторії», натиснути кнопку «Додати аудиторію», та у модальному вікні додати аудиторію (приклад наведено на рис. 5.23). Також оператор-адмін може передивитися розклад (приклад наведено на рис. 5.24), для додавання аудиторій, оператор повинен перейти на підсторінку «Створити розклад», після цього заповнити розклад по групі (приклад наведено на рис. 5.25).

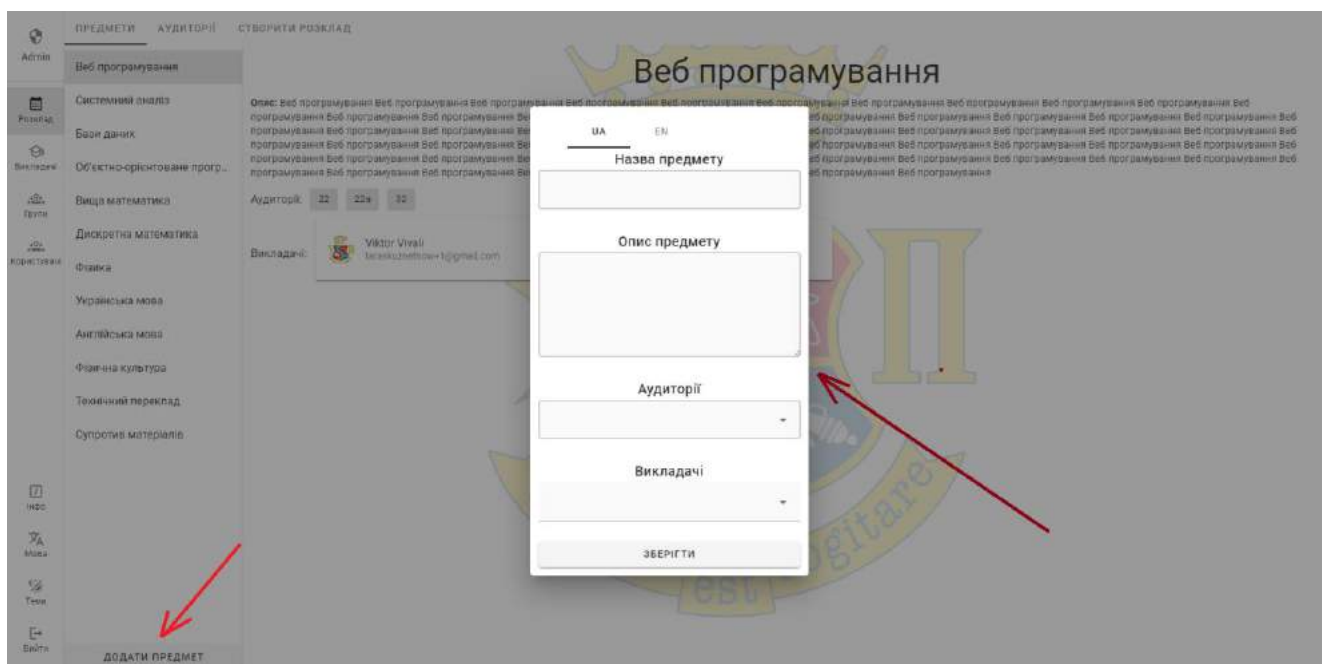


Рисунок 5.22 – Модальне вікно додавання предметів

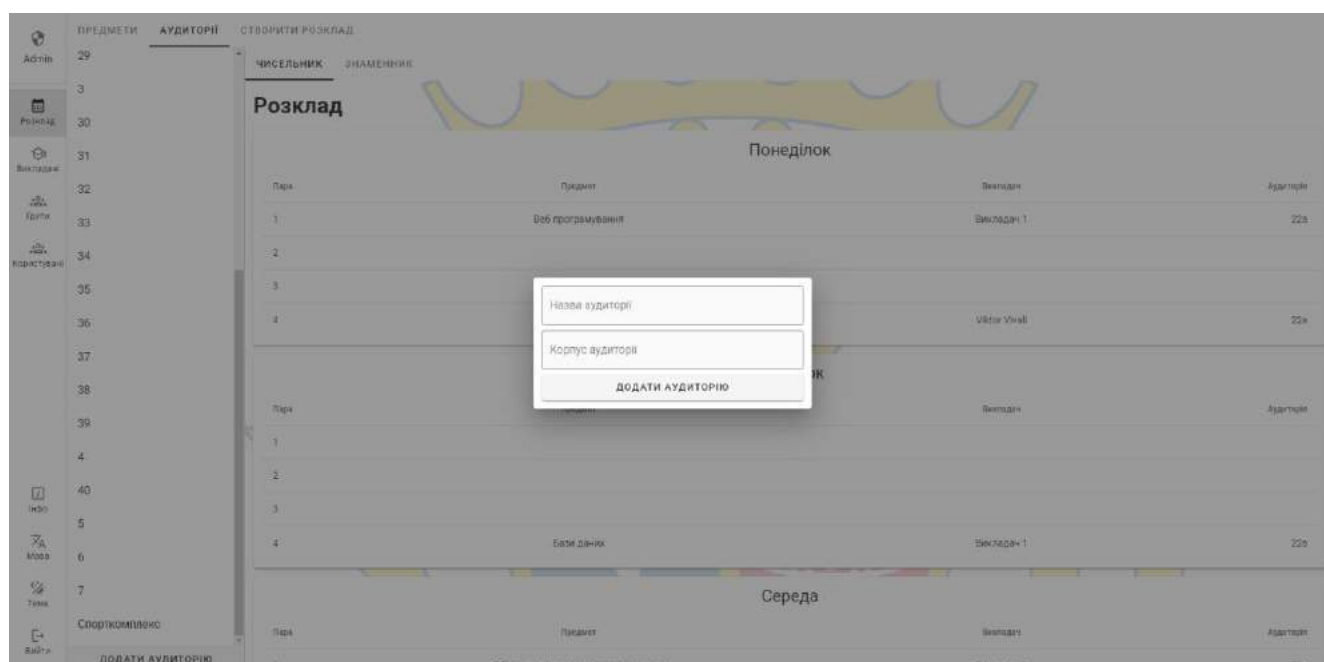


Рисунок 5.23 – Модальне вікно додавання аудиторії

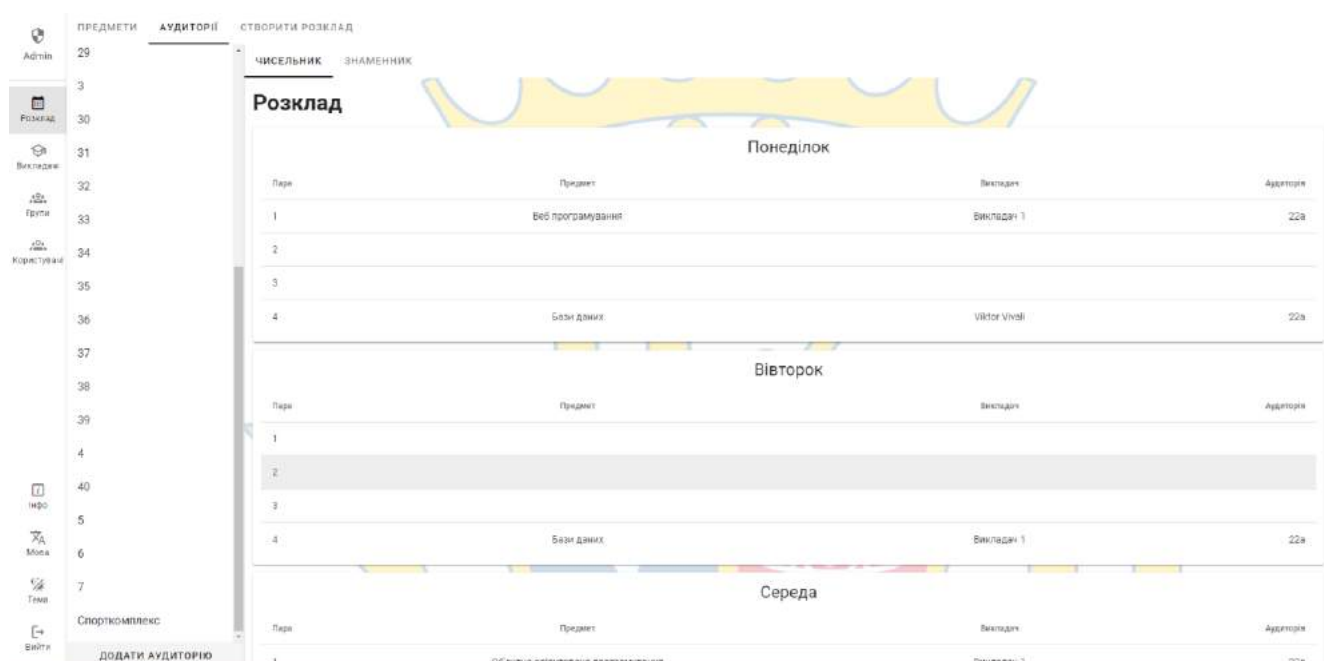


Рисунок 5.24 – Сторінка перегляду розкладу за аудиторіями

Рисунок 5.25 – Сторінка додавання розкладу оператором

5.4 Повідомлення оператору

В ході роботи оператор може отримати такі повідомлення, описи яких міститься в таблиці 5.1.

Таблиця 5.1 — Повідомлення та їхні описи

Повідомлення українською	Повідомлення англійською	Опис
Неправильний логін чи пароль	Incorrect login or password	Оператор ввів неправильний логін чи пароль під час авторизації
Користувач із такою електронною поштою вже існує в системі	User with the same email are already exist in the system	Оператор ввів електронну пошту, яка вже зареєстрована в системі
Користувач з таким ж ідентифікатором вже існує	User with same userName is already exist	Оператор ввів електронну адресу, яка вже зареєстрована в системі

Продовження таблиці 5.1

Повідомлення українською	Повідомлення англійською	Опис
Така група вже існує	Group is already exist	Оператор ввів назву групи, яка вже є у системі
Ім'я групи не може бути порожнім	Group name can not be empty	Оператор ввів пусту назву групи
Назва групи не може складатися менше ніж з трьох літер	Group name can not be less then three letter	Оператор ввів назву групи, яка складається менш ніж з трьох символів
Необхідно ввести пароль	Password Required	Оператор не заповнив поле пароллю
Електронна пошта обов'язкова	E-mail is required	Оператор не заповнив поле електронної пошти
Обов'язковий пункт	Item is required	Оператор не заповнив один з обов'язкових пунктів

5.3 Висновки по розділу

На даному етапі роботи було складено «Керівництво оператора», що містить призначення і умови виконання програми, описано процес виконання програми, а також приведено перелік повідомлень та їхні описи.

6 ТЕСТУВАННЯ І РЕЗУЛЬТАТИ

Цей розділ описує обрану методологію тестування програмного забезпечення та результати тестів, які перевіряють відповідність програмного забезпечення завданням, зазначеним у специфікації проєкту.

6.1 Опис методики проведення тестування

Клієнтські та серверні частини були протестовані вручну. Таким чином, можливо оцінити якість макета вебзастосунку без написання складної логіки для автоматичного тестування графічного інтерфейсу користувача.

6.2 Результати тестування

Тестування на стороні клієнта проводилося вручну. Використовуючи інструменти розробника, вбудовані в браузер Google Chrome, адаптивність макета вебпрограми та зовнішнього вигляду вебзастосунку Graphics I при зміні ширини екрану контролювалася як з мобільних пристроїв, так і з планшетів і комп'ютерів.

Також була перевірена реакція системи на неправильне введення даних, а також неправильне введення даних і поведінку в разі помилок перевірки. Процес тестування був наступним:

- якщо під час авторизації в систему, користувач введе логін та пароль яких немає в базі, буде отримане повідомлення «Неправильний логін чи пароль» (див. рис 6.1);
- якщо під час введення даних у поля реєстрації, будуть введені невалідні дані, помилки будуть виведені під кожним невалідним полем (див. рис 6.2);
- якщо під час введення даних у поля реєстрації, ввести адресу електронної пошти, який все є у системі, буде отримане повідомлення «Користувач із такою електронною поштою вже існує в системі» (див. рис 6.3);

- якщо під час зміни даних користувача, користувач введе електронну адресу, який вже існує в базі, буде отримане повідомлення «Користувач з таким ж ідентифікатором вже існує» (див.рис 6.4);
- якщо під час додавання нової групи, ввести номер групи, який вже існує у системі, буде отримане повідомлення «Така група вже існує» (див. рис. 6.5);
- якщо під час додавання нової групи, ввести номер групи, який буде менший за 3 символи, буде отримане повідомлення «Назва групи не може складатися менше ніж з трьох літер» (див. рис. 6.6);
- якщо під час додавання нової групи, назву групи залишити пустою, буде отримане повідомлення «Ім'я групи не може бути порожнім» (див. рис. 6.7);
- якщо під час додавання нової аудиторії, корпус аудиторії залишити пустою, буде отримане повідомлення «Корпус не може бути порожнім» (див. рис. 6.8);
- якщо під час додавання нової аудиторії, назву аудиторії залишити пустою, буде отримане повідомлення «Аудиторія не може бути порожньою» (див. рис. 6.9);
- якщо під час додавання нового предмету, назву предмету залишити пустою, буде отримане повідомлення «Неможливо зберегти предмет без назви» (див. рис. 6.10);
- якщо під час додавання нового предмету, ні вибрати не однієї аудиторії, буде отримане повідомлення «Будь ласка, оберіть будь-яку аудиторію» (див. рис. 6.11);
- якщо під час додавання нового предмету, не вибрати ні одного викладача, буде отримане повідомлення «Будь ласка, оберіть будь-якого викладача» (див. рис. 6.12).

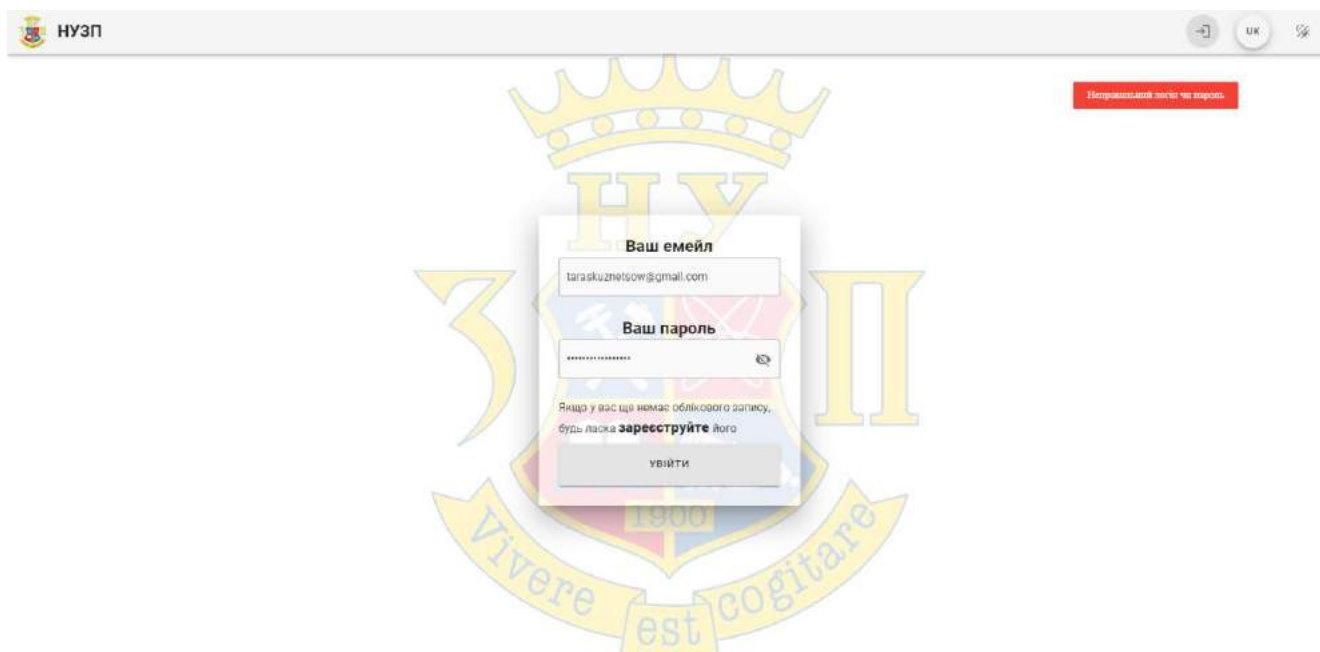


Рисунок 6.1 – Реакція системи на помилку під час заповнення форми авторизації

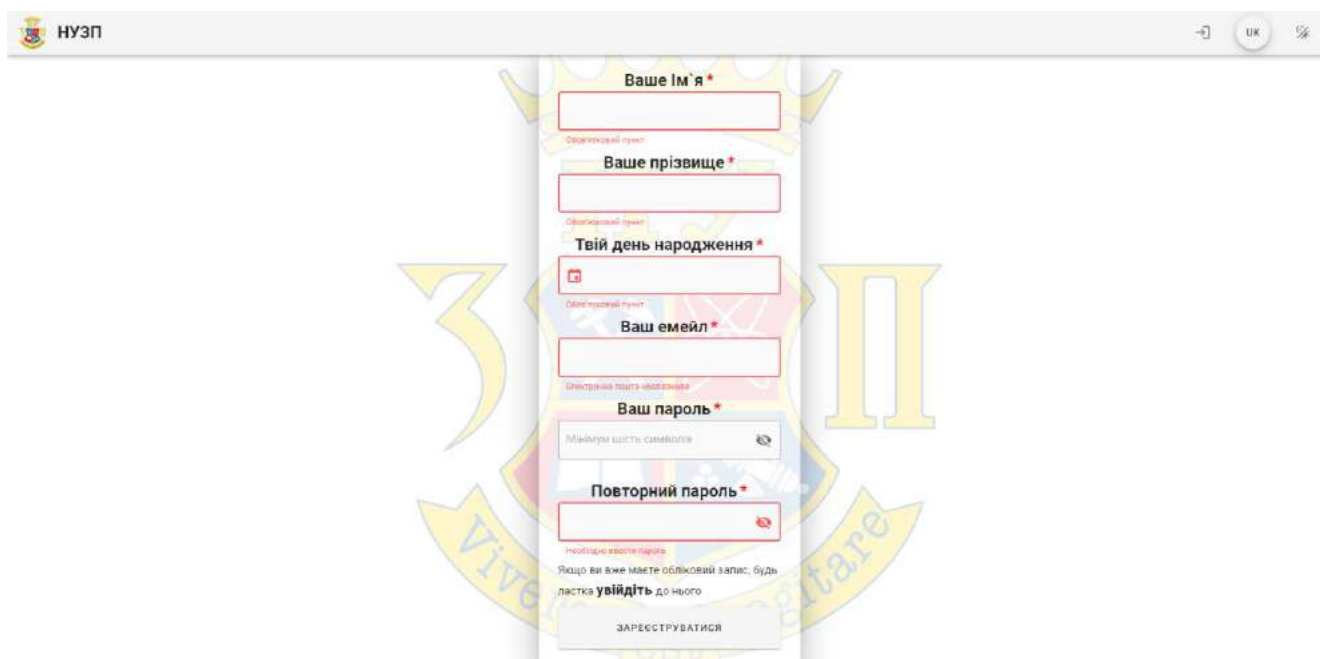


Рисунок 6.2 – Реакція системи на помилку під час заповнення форми реєстрації

Ваше ім'я*

Тарас

Ваше прізвище*

sdsdsds

Твій день народження*

2023-12-08

Ваш емейл*

taraskuznetsow@gmail.com

Ваш пароль*

Повторний пароль*

Якщо ви вже маєте обліковий запис, будь ласка **увійдіть** до нього

ЗАРЕЄСТРУВАТИСЯ

Використовуйте тільки електронну пошту як адресу системи

Рисунок 6.3 – Реакція системи на введення електронної адреси користувача при реєстрації, яка вже існує в системі

Профіль

Фото

Ваше ім'я

Тарас

Ваше прізвище

Kuznetsov

Твій день народження

2000-05-24

Ваш емейл

taraskuznetsow@gmail.com

@ Ваш юзернейм

@str1m1tch1

ЗБЕРІГТИ

Використовуйте тільки цілком контрольований вай-фай

Рисунок 6.4 – Реакція на вже існуючий ідентифікатор користувача в системі

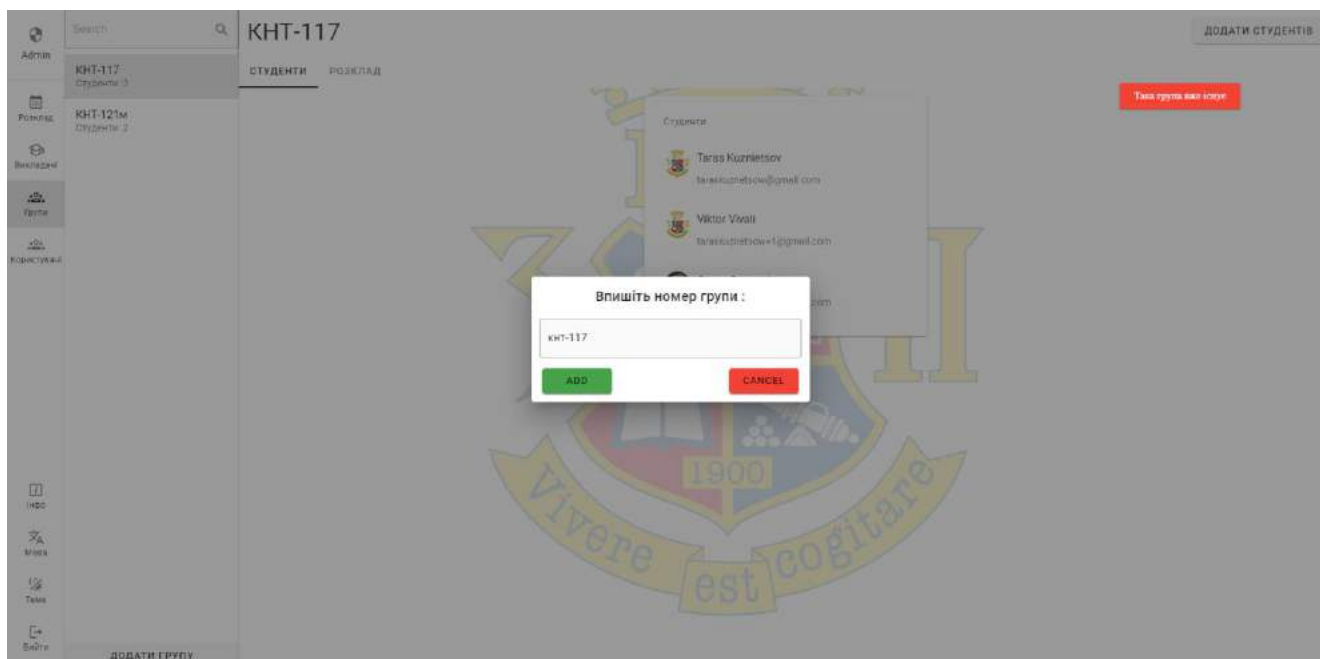


Рисунок 6.5 – Реакція на вже існуючу групу у системі

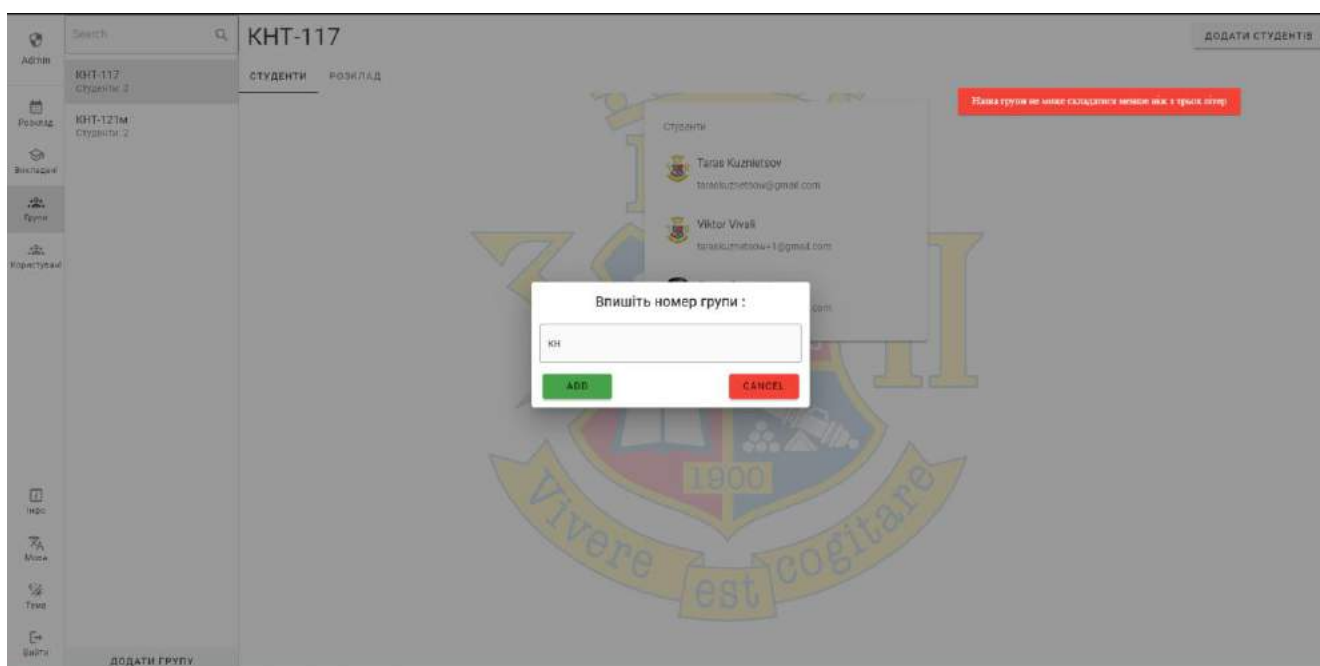


Рисунок 6.6 – Реакція системи на недостатню кількість символів у назві групи

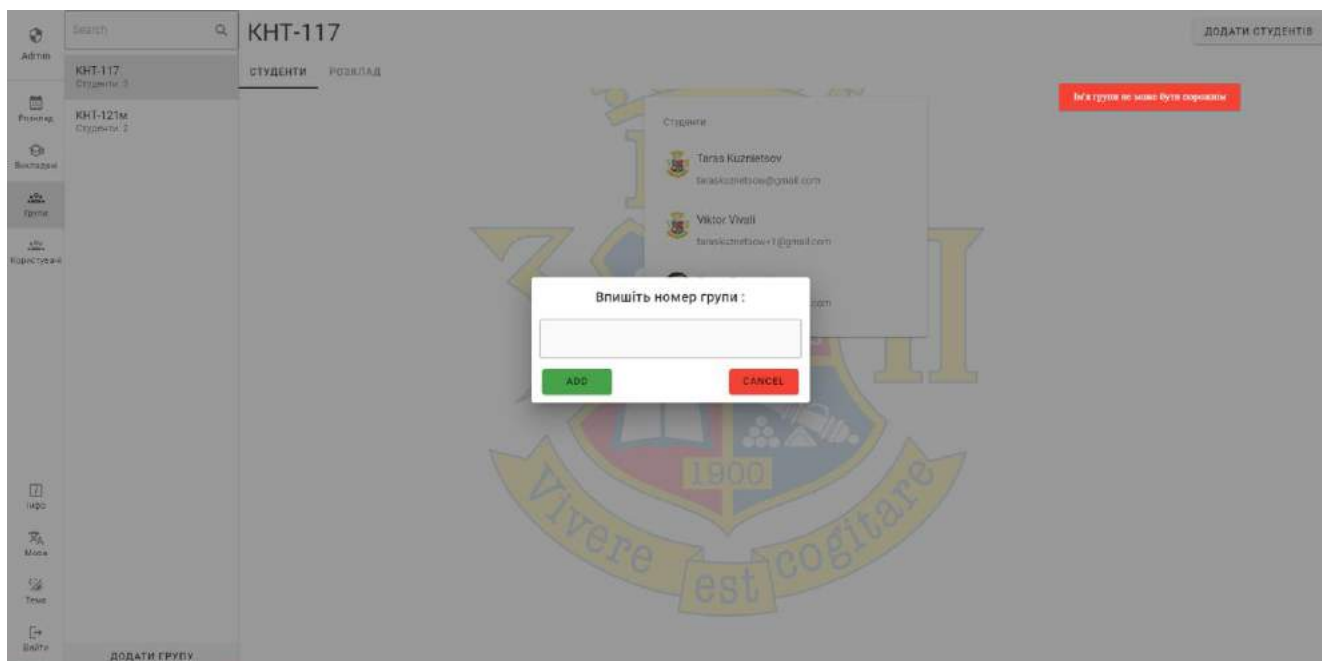


Рисунок 6.7 – Реакція системи на пусту назву групи при її додаванні

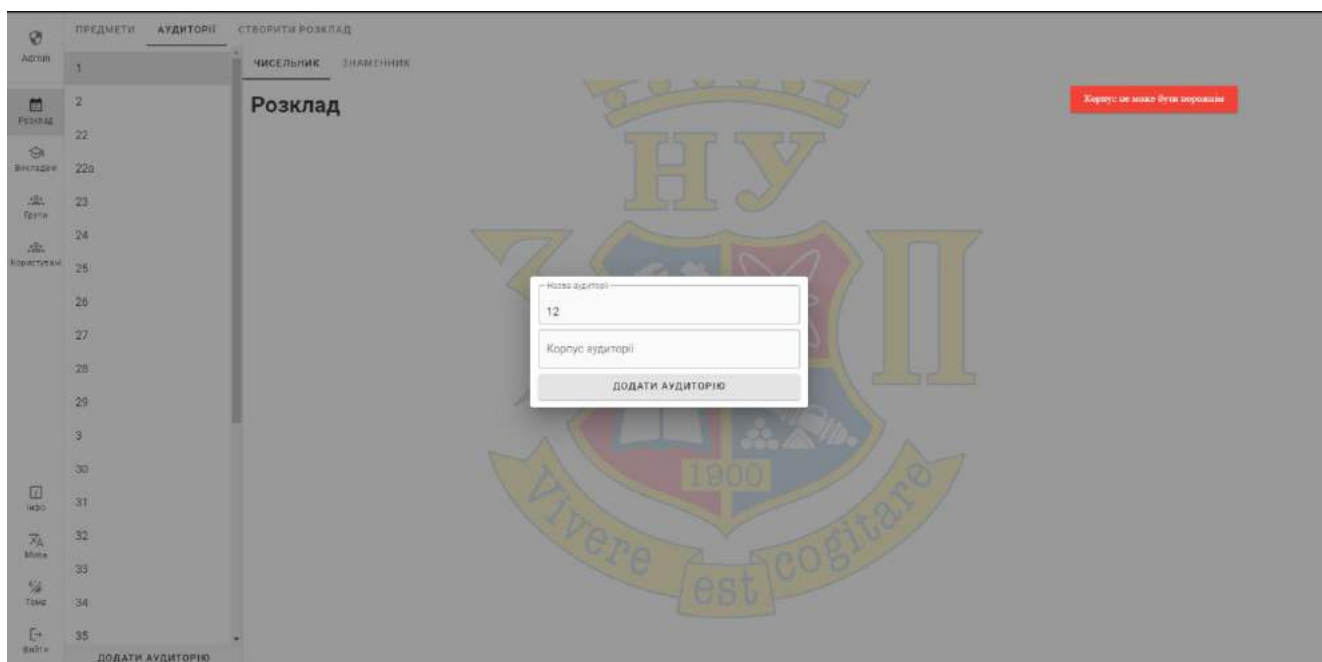


Рисунок 6.8 – Реакція системи на пусту назву корпусу аудиторії при її додаванні

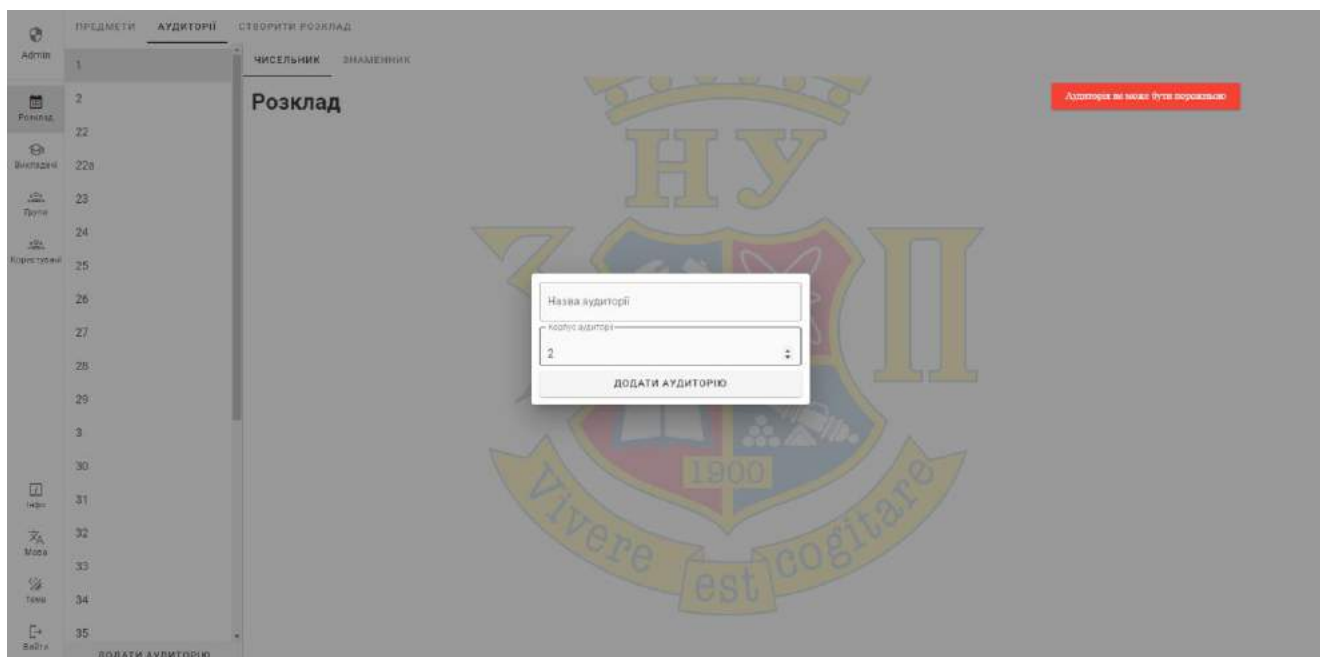


Рисунок 6.9 – Реакція системи на пусту назву аудиторії при її додаванні

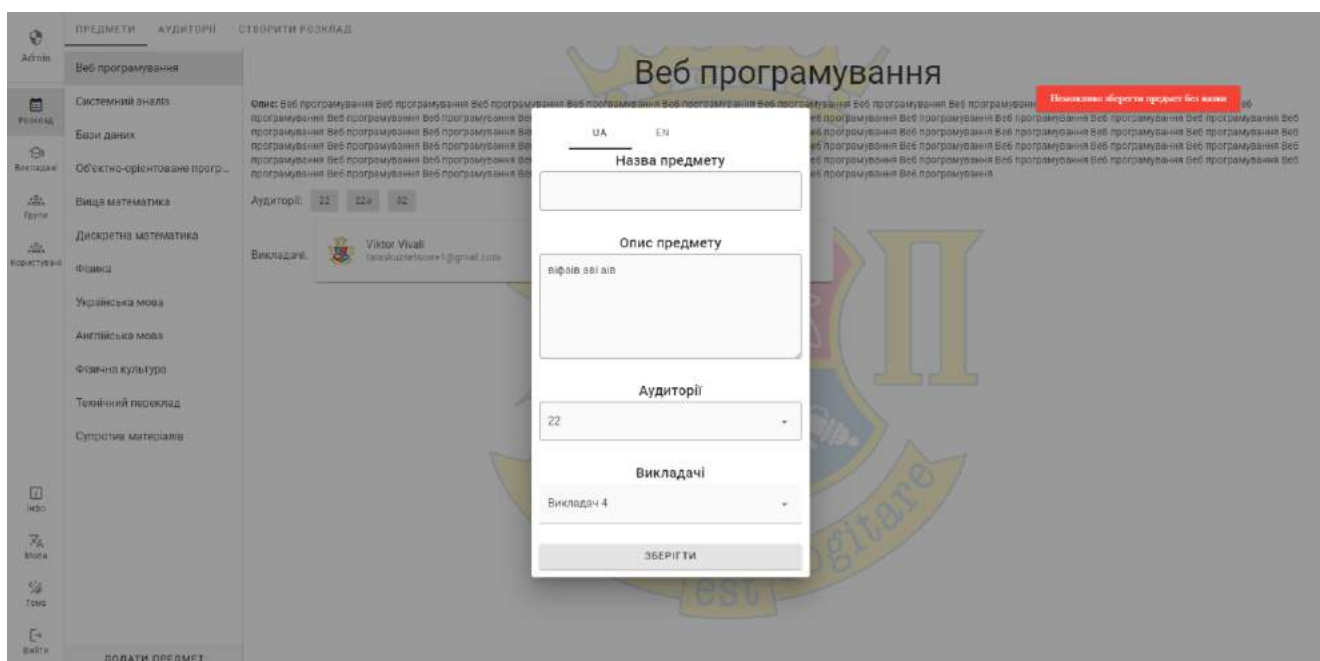


Рисунок 6.10 – Реакція системи на пусту назву предмету при його додаванні

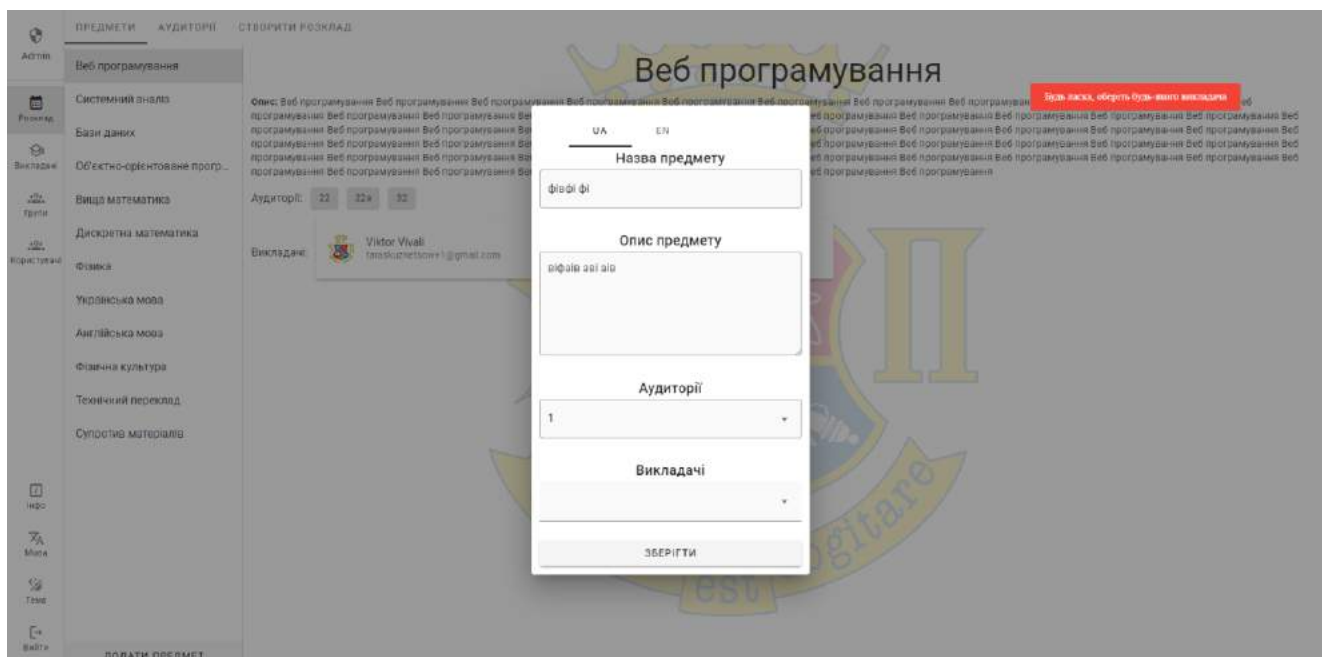


Рисунок 6.11 – Реакція системи на не вибраних викладачів при додаванні предмету

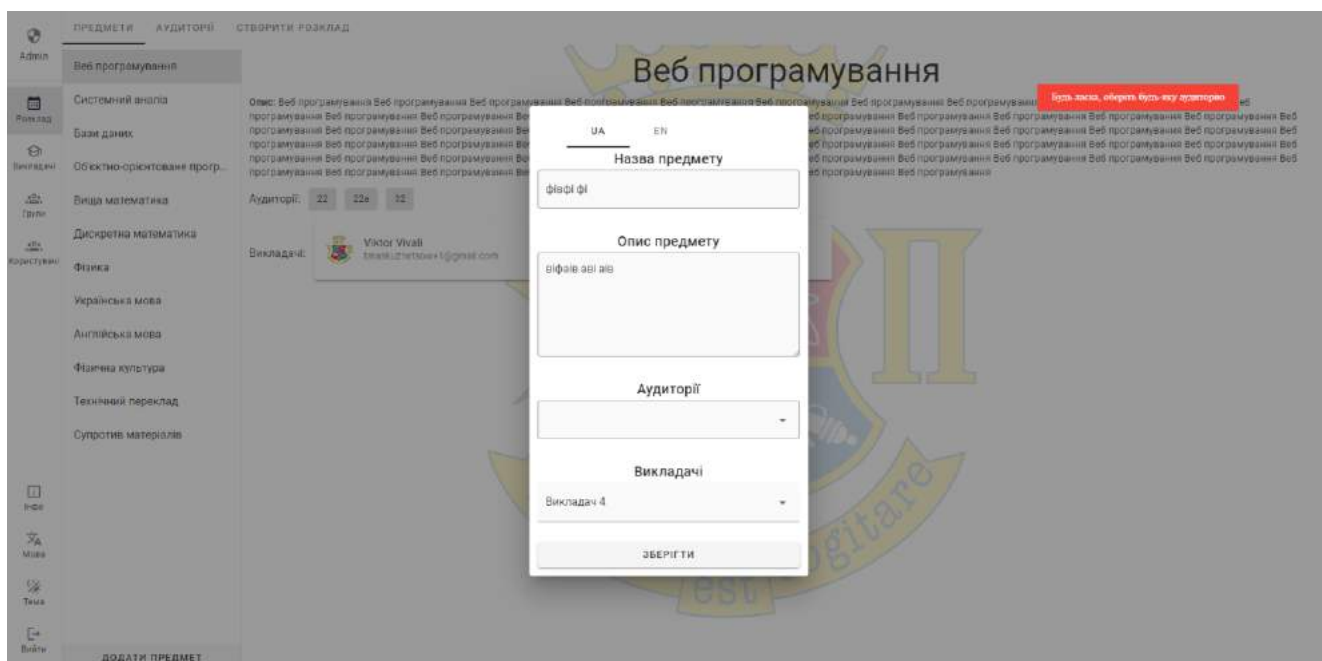


Рисунок 6.12 – Реакція системи на не вибрані аудиторії при додаванні предмету

ВИСНОВКИ

Під час виконання роботи було досліджено предметну область, розглянуто проблематику поставленого завдання, наведено його актуальність, доцільність. В даній роботі вивчалася проблема покращення навчального процесу закладів вищої освіти.

На першому етапі роботи було ретельно проаналізовано предметну область: сформовано проблему та актуальність роботи, розглянуто існуючі методи вирішення розв'язуваної проблеми та готові рішення, виявлено їх переваги та недоліки. Наступним кроком стала розробка архітектури системи, в ході якої було описано структуру розроблюваної програми, обрано інструменти для розробки ПЗ, а також спроектовано майбутню БД. На наступному етапі було реалізовано компоненти програми на основі «Технічного завдання» та надано їхнє обґрунтування.

Мета роботи було створення вебзастосунку для покращення навчального процесу ЗВО з інтелектуальним чатом.

Було проведено порівняльний аналіз основних методів для вирішення цього завдання та встановлено, що існуючі засоби не відповідають повному функціоналу, якій потрібен для вирішення завдання дипломної роботи. Тому виникає необхідність в реалізації нового вебзастосунку з розширеним функціоналом.

Відповідно до мети були поставлені і вирішені наступні завдання:

- реалізувати зручну авторизація користувача;
- реалізувати зручну реєстрація користувачів;
- реалізувати зручне створення чатів;
- реалізувати зручний перегляд студентів, викладачів, груп, аудиторій, предметів та розкладу;
- реалізувати зручний мультимовний інтерфейс;
- реалізувати можливість зручного листування користувачів.

Розроблено вебзастосунок для зручної комунікації між учасниками навчального процесу (викладачами та студентами). Серверна частина створеного ПП було написаний на мові програмування C#. Клієнтська частина створеного ПП була написана на мові програмування TS. У якості середовища розробки клієнтської частини було використано Visual studio code, а для серверної Visual studio. Для роботи з базою даних була використана бібліотека Entity Framework.

В результаті роботи було отримано програмний продукт, що дозволяє виконати поставлене завдання, а саме – покращення навчального процесу. Роботу програми було протестоване на тестових даних. За необхідності програма може бути модифікована та використана під час розробки більш складної, комплексної системи, що вирішуватиме завдання покращення навчального процесу у ЗВО НУ «Запорізька політехніка».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Blackboard Learn [Electronic resource]. – Access mode:
<https://www.blackboard.com/>
2. Schoology [Electronic resource]. – Access mode:
<https://www.powerschool.com/>
3. Edmodo [Electronic resource]. – Access mode:
<https://www.edmodo.com/>
4. JavaScript [Електронний ресурс]. – Режим доступу:
<https://uk.wikipedia.org/wiki/JavaScript>
5. TypeScript [Електронний ресурс]. – Режим доступу:
<https://uk.wikipedia.org/wiki/TypeScript>
6. Python [Електронний ресурс]. – Режим доступу:
<https://uk.wikipedia.org/wiki/Python>
7. PHP [Електронний ресурс]. – Режим доступу:
<https://uk.wikipedia.org/wiki/PHP>
8. Vue.Js [Електронний ресурс]. – Режим доступу:
<https://uk.vuejs.org/v2/guide/index.html>
9. React [Електронний ресурс]. – Режим доступу:
<https://uk.wikipedia.org/wiki/React>
10. Angular [Електронний ресурс]. – Режим доступу:
<https://apeirondb.com/ua/blog/it-technologies-angular>
11. SPA [Electronic resource]. – Access mode:
https://en.wikipedia.org/wiki/Single-page_application.com
12. Knex.js [Electronic resource]. – Access mode: <http://knexjs.org/>
13. MYSQL [Електронний ресурс]. – Режим доступу:
<https://uk.wikipedia.org/wiki/MySQL>
14. Microsoft SQL Server [Електронний ресурс]. – Режим доступу:
https://uk.wikipedia.org/wiki/Microsoft_SQL_Server
15. Entity Framework [Електронний ресурс]. – Режим доступу:

https://uk.wikipedia.org/wiki/Entity_Framework

16. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа:
https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio

ДОДАТОК А
Технічне завдання

ВСТУП

Розроблений програмний продукт використовується для вирішення проблеми вдосконалення навчального процесу у вищих навчальних закладах. Одним з основних завдань першого етапу створення раціональної системи смарт-чату є створення логічного та зрозумілого дизайну.

Програмне забезпечення соціальної платформи NUZP Social забезпечує моніторинг програми з різних цифрових пристроїв, таких як комп'ютери, планшети та смартфони. Програмне забезпечення дозволяє користувачам відстежувати розклад дзвінків, розклад різних навчальних груп, спілкування з іншими студентами та викладачами, додавання предметів, аудиторій, ролей, груп та розкладів.

A.1 Мета розробки і її призначення

Об'єкт дослідження – програмні засоби для реалізації інтелектуального чату та систем для покращення навчального процесу закладів вищої освіти (ЗВО).

Предмет дослідження – вебзастосунок інтелектуального чату для покращення навчального процесу ЗВО.

Мета роботи – створення вебзастосунку інтелектуального чату та додаткових систем, необхідних для покращення навчального процесу.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування JavaScript, мова програмування C#, мова програмування TypeScript, система для побудови клієнтських вебзастосунків, середовище розробки Visual Studio Code, Visual Studio персональний комп'ютер з процесором AMD Ryzen 5 під управлінням операційної системи Linux Ubuntu 20.04.

Результати. Створено програмний вебзастосунок, який виконує обробку даних, необхідних для покращення навчального процесу закладу вищої освіти.

Висновки. Розроблено інтелектуальний вебзастосунок для покращення навчального процесу закладів вищої освіти, необхідний для зручного користування студентами та викладачами університету.

Галузь використання – ЗВО.

А.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

При розробці інтелектуального чату для студентів з метою покращення навчального процесу постає завдання відбору раціональних даних для відображення на сайті. Відбір раціональних даних означає правильну побудову запитів до бази даних для отримання даних, необхідних для конкретного викладача або студента.

А.3 Підстави для розробки

Підставою для виконання розробки є завдання на дипломний проєкт на тему “Дослідження та програмна реалізація інтелектуального сайту-чату для студентів: покращення навчального процесу”, затверджене наказом по НУ «Запорізька політехніка» № 437 від 14 листопада 2023 року.

А.4 Основні вимоги до програми, що розробляється, початковим даним і результату

А.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- повинен бути реалізований адаптивний графічний інтерфейс користувача;

- повинна бути реалізована можливість вибору студента, викладача або групи з БД;

- повинна бути реалізована можливість пошуку розкладу студентів НУ «Запорізька політехніка»;
- повинна бути реалізована можливість переведення інтерфейсу вебзастосунку на англійську мову;
- повинна бути реалізована можливість спілкування з іншими користувачами-студентами або ж викладачами;
- повинна бути реалізована сторінка для перегляду списку груп у ЗВО, а також списку студентів за групами, які надали цей доступ;
- повинна бути реалізована сторінка для перегляду розкладу дзвінків у ЗВО.

A.4.2 Вимоги до надійності

ПП повинна виконувати наступні вимоги до надійності:

- захищати ПП від спроби завантаження непотрібних даних до БД;
- захищати ПП від помилкових дій користувача (наприклад користувач ввів невалідні дані);
- обробляти виняткові ситуації (наприклад, відсутність підключення до бази даних);

ПП повинен нормально функціонувати при безперервній роботі комп'ютера. У разі виникнення збоїв в роботі серверу, відновлення нормальної роботи ПП повинно проводитися після перезавантаження системи.

A.4.3 Умови експлуатації

Умови експлуатації:

- програмне забезпечення можна запуснути на різних серверах, за допомогою клонування з репозиторію;
- для користування програмним забезпеченням не потрібна додаткова кваліфікація та додаткові навички;

– програмне забезпечення повинне мати адаптивну верстку для пристроїв, з яких можна запустити сучасний браузер.

A.4.4 Вимоги до складу та параметрів технічних засобів

Вимоги до складу та параметрів технічних засобів:

- платформа x64-based;
- операційна система Microsoft Windows 10/8/7 , Linux Ubuntu 16.04 та інші аналоги сучасних UNIX-оїдних систем;
- процесор з тактовою частотою не нижче 1.4 ГГц;
- 1 ГБ оперативної пам'яті, рекомендується 8 ГБ оперативної пам'яті для швидкого реагування інтерфейсу на різні запити;
- вільне місце на диску 500 МБ мінімум.

A.4.5 Вимоги до маркування та упакування

Програмне забезпечення не вимагає жодних додаткових кроків для етикетування або пакування.

A.4.6 Вимоги до транспортування та збереження

Вимоги до транспортування та зберігання подібні до вимог до носіїв, на яких записані програми.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- керівництво програміста;
- керівництво оператора;

– результати тестування.

А.6 Основні обмеження на установку і використання програмного продукту

Програмні продукти, що розробляються, повинні працювати на сервері і вносити певні зміни в конфігурацію проєкту. Програмний продукт повинен використовуватися на комп'ютері, який відповідає апаратним вимогам і параметрам. Програмний продукт може використовуватися користувачем у будь-який час за умови підключення до мережі Інтернет.

ДОДАТОК Б
Текст програми

Б.1 Файл Program.cs

```

using CORE.Constants;
using CORE.Helpers;
using DAL;
using Diploma_server.Hubs;
using Diploma_server.Infrastructure.Middleware.Extensions;
using Microsoft.AspNetCore.Authentication.Certificate;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Localization;
using Microsoft.EntityFrameworkCore;

namespace Diploma_server
{
    public class Program
    {
        public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);
            builder.Services.AddApplicationIdentityServices();
            builder.Services.AddApplicationInternalServices();
            builder.Services.AddApplicationInternalRepositories();
            builder.Services.AddApplicationAuthentication();

            builder.Services.AddApplicationCorsPolicy();

            builder.Services.AddControllers();

            builder.Services.AddSignalR(options =>
            {
                options.ClientTimeoutInterval = TimeSpan.FromSeconds(500);
            });

            string connection = builder.Configuration.GetConnectionString("DefaultConnection");
            builder.Services.AddDbContext<ApplicationContext>(options =>
            options.UseSqlServer(connection));

            builder.Services.AddAuthentication(options =>
            {
                options.DefaultAuthenticateScheme = CertificateAuthenticationDefaults.AuthenticationScheme;
                options.DefaultChallengeScheme = CertificateAuthenticationDefaults.AuthenticationScheme;
            }).AddJwtBearer("Custom", options =>
            {
                options.Authority = "http://192.168.0.166:3000/";
                options.Audience = "http://192.168.0.166:3000/";
                options.Events = new JwtBearerEvents
                {
                    OnMessageReceived = context =>
                    {
                        var accessToken = context.Request.Query["access_token"];

                        // If the request is for our hub...
                        var path = context.HttpContext.Request.Path;
                        if (!string.IsNullOrEmpty(accessToken) &&
                            (path.StartsWithSegments("/hubs/chat")))
                        {
                            // Read the token out of the query string
                            context.Token = accessToken;
                        }
                    }
                }
            });
        }
    }
}

```

```

        return Task.CompletedTask;
    }
};
});

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();
app.UseCors(ApplicationConstants.APIOptions.CorsPolicy);

app.UseRequestLocalization(new RequestLocalizationOptions
{
    ApplyCurrentCultureToResponseHeaders = true,
    DefaultRequestCulture = new RequestCulture(LocalizationConstants.UkraineCultureCode),
    SupportedCultures = LocalizationHelper.GetAvailableLocalizations(),
    SupportedUICultures = LocalizationHelper.GetAvailableLocalizations(),
});

// Configure the HTTP request pipeline.
//if (app.Environment.IsDevelopment())
//{
    app.UseSwagger();
    app.UseSwaggerUI();
//}

app.UseHttpsRedirection();
app.UseCertificateForwarding();

app.UseAuthorization();
app.UseAuthentication();

app.MapHub<Chat>($"/{nameof(Chat)}");

app.MapControllers();

app.Run();
}
}
}

```

Б.2 Файл AuditoryController.cs

```

using BLL.Interfaces;
using Microsoft.AspNetCore.Mvc;
using Repositories.DTOs.Auditories;

namespace Diploma_server.Controllers
{
    [ApiController]
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/auditory/")]
    public class AuditoryController : ControllerBase
    {

```

```

private readonly IAuditoryService _service;

public AuditoryController(IAuditoryService service)
{
    _service = service;
}

[HttpPost]
[Route("addNewAuditory")]
[ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<IActionResult> AddNewAuditory(AddAuditoryDTO dto)
{
    try
    {
        await _service.AddNewAuditoryAsync(dto);

        return Ok();
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[HttpGet]
[Route("getAllAuditories")]
[ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<IActionResult> GetAllAuditories()
{
    try
    {
        return Ok(await _service.GetAllAuditoriesAsync());
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

```

```

    }
}

[HttpDelete]
[Route("removeAuditory")]
[ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<IActionResult> RemoveAuditory([FromBody] AuditoryShortDTO dto)
{
    try
    {
        await _service.DeleteAuditoryAsync(dto);

        return Ok();
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
}
}

```

Б.3 Файл ChatController.cs

```

using BLL.Interfaces;
using Diploma_server.Infrastructure.Extensions;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Repositories.DTOs.Chat;

namespace Diploma_server.Controllers
{
    [ApiController]
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/[controller]/")]
    [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
    public class ChatController : ControllerBase
    {
        private readonly IChatService _chatService;

        public ChatController(IChatService chatService)
        {
            _chatService = chatService;
        }
    }
}

```

```

    }

    [HttpPost]
    [Route("createNewSingleChatAndGetNewChatId")]
    public async Task<IActionResult> CreateNewSingleChatAndGetNewChatId([FromQuery] int userId)
    {
        try
        {
            var currentUserId = User.GetUserIdOrDefault();

            return Ok( await _chatService.GetOrCreateNewSingleChatAndGetNewChatId(currentUserId.Value, userId) );
        }
        catch(Exception e)
        {
            return BadRequest(e);
        }
    }

    [HttpGet]
    [Route("getAllUsersChatsAsync")]
    public async Task<IActionResult> GetAllUsersChatsAsync()
    {
        try
        {
            var currentUserId = User.GetUserIdOrDefault();

            return Ok(await _chatService.GetAllUsersCahtsAsync(currentUserId.Value));
        }
        catch(Exception e)
        {
            return BadRequest(e.Message);
        }
    }

    [HttpGet]
    [Route("getAllChatMessages")]
    public async Task<IActionResult> GetAllChatMessages([FromQuery] int chatId)
    {
        try
        {
            return Ok(await _chatService.GetAllChatMessagesAsync(chatId));
        }
        catch (Exception e)
        {
            return BadRequest(e.Message);
        }
    }
}

```

Б.4 Файл AuthController.cs

```

using BLL.Interfaces;
using CORE.Enums;
using Diploma_server.Models.UserProfile;
using Microsoft.AspNetCore.Mvc;
using Repositories.DTOs.UserProfile;

namespace Diploma_server.Controllers
{

```

```

[ApiController]
[ApiConventionType(typeof(DefaultApiConventions))]
[Route("api/auth/")]
public class AuthController : ControllerBase
{
    private readonly IAuthorizationService _service;

    public AuthController(IAuthorizationService service)
    {
        _service = service;
    }

    [HttpPost]
    [Route("createRole")]
    [ProducesResponseType(typeof(bool), StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status400BadRequest)]
    [ProducesResponseType(StatusCodes.Status401Unauthorized)]
    [ProducesResponseType(StatusCodes.Status500InternalServerError)]
    public async Task<IActionResult> CreateRole([FromQuery] ERoleEnum role)
    {
        try
        {
            await _service.CreateNewRoleAsync(role);

            return Ok(true);
        }
        catch (Exception e)
        {
            return BadRequest(e.Message);
        }
    }

    [HttpPost]
    [Route("registration")]
    [ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status400BadRequest)]
    [ProducesResponseType(StatusCodes.Status401Unauthorized)]
    [ProducesResponseType(StatusCodes.Status500InternalServerError)]
    public async Task<IActionResult> RegistrationUser([FromBody] RegistrationModel model)
    {
        try
        {
            var token = await _service.RegisterNewUserAsync(new RegistrationDTO
            {
                Email = model.Email,
                BirthDate = model.BirthDate,
                Password = model.Password,
                Surname = model.Surname,
                Name = model.Name
            });

            return Ok(token);
        }
        catch (Exception e)
        {
            return BadRequest(e.Message);
        }
    }

    [HttpPost]
    [Route("login")]
    [ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status400BadRequest)]

```

```

[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
//[RequireAccess(ApiKeyAccessType.Read)]
public async Task<IActionResult> Login(LoginModel model)
{
    try
    {
        var token = await _service.LoginAsync(model.Email, model.Password);

        return Ok(token);
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}
}
}

```

Б.5 Файл GroupsController.cs

```

using BLL.Interfaces;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace Diploma_server.Controllers
{
    [ApiController]
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/[controller]/")]
    [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
    public class GroupsController : ControllerBase
    {

        private readonly IGroupService _service;

        public GroupsController(IGroupService service)
        {
            _service = service;
        }

        [HttpGet]
        [Route("getAllGroups")]
        public async Task<IActionResult> GetAllGroups()
        {
            try
            {
                return Ok(await _service.GetAllGroupsAsync());
            }
            catch (Exception e)
            {
                return BadRequest(e);
            }
        }

        [HttpPost]
        [Route("addNewGroup")]
        public async Task<IActionResult> AddNewGroup([FromQuery] string? groupName)
        {
            try

```

```

        {
            await _service.AddNewGroupAsync(groupName);
            return Ok();
        }
        catch(Exception e)
        {
            return BadRequest(e.Message);
        }
    }

    [HttpGet]
    [Route("getGroupById")]
    public async Task<IActionResult> GetGroupById([FromQuery] int groupId)
    {
        try
        {
            return Ok(await _service.GetGroupsByIdAsync(groupId));
        }
        catch(Exception e)
        {
            return BadRequest(e.Message);
        }
    }
}
}
}

```

Б.6 Файл ImageController.cs

```

using BLL.Interfaces;
using CloudinaryAPI.Enums;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Repositories.Interface;

namespace Diploma_server.Controllers
{
    [ApiController]
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/[controller]/")]
    [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
    public class ImageController : ControllerBase
    {
        private readonly IImageService _service;
        public ImageController( IImageService imageService )
        {
            _service = imageService;
        }

        [HttpPost]
        [Route("UploadUserProfilePhoto")]
        [ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        [ProducesResponseType(StatusCodes.Status401Unauthorized)]
        [ProducesResponseType(StatusCodes.Status500InternalServerError)]
        public async Task<IActionResult> UploadUserProfilePhoto(IFormFileCollection files, [FromQuery]
        ECloudinaryFolders type)
        {
            try
            {
                var file = files[0];
            }
        }
    }
}

```

```

        return Ok(await _service.UploadUserProfileImage(file, type));
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
}
}

```

B.7 ScheduleController.cs

```

using BLL.Interfaces;
using Diploma_server.Infrastructure.Extensions;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Repositories.DTOs.Schedule;

namespace Diploma_server.Controllers
{
    [ApiController]
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/schedule/")]
    [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
    public class ScheduleController : ControllerBase
    {
        private readonly IScheduleService _service;

        public ScheduleController(IScheduleService service)
        {
            _service = service;
        }

        [HttpPost]
        [Route("createSchedule")]
        [ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        [ProducesResponseType(StatusCodes.Status401Unauthorized)]
        [ProducesResponseType(StatusCodes.Status500InternalServerError)]
        public async Task<IActionResult> CreateSchedule([FromBody] CreateScheduleDTO dto)
        {
            try
            {
                await _service.AddScheduleAsync(dto);
                return Ok();
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [HttpPost]
        [Route("getScheduleByGroupId")]
        [ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        [ProducesResponseType(StatusCodes.Status401Unauthorized)]
        [ProducesResponseType(StatusCodes.Status500InternalServerError)]
    }
}

```

```

public async Task<IActionResult> GetScheduleByGroupId([FromBody] ScheduleFilterDTO filter)
{
    try
    {
        if(filter.IsFindByAuthUser)
        {
            var currentUserId = User.GetUserIdOrDefault();
        }

        return Ok(await _service.GetScheduleAsync(filter));
    }
    catch(Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
}
}

```

Б.8 Файл SubjectController.cs

```

using BLL.Interfaces;
using Microsoft.AspNetCore.Mvc;
using Repositories.DTOs.Auditories;
using Repositories.DTOs.Subjects;

namespace Diploma_server.Controllers
{
    [ApiController]
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/subject")]
    public class SubjectController : ControllerBase
    {
        private readonly ISubjectService _service;

        public SubjectController(ISubjectService service)
        {
            _service = service;
        }

        [HttpPost]
        [Route("addNewSubject")]
        [ProducesResponseType(typeof(string), StatusCodes.Status200OK)]
        [ProducesResponseType(StatusCodes.Status400BadRequest)]
        [ProducesResponseType(StatusCodes.Status401Unauthorized)]
        [ProducesResponseType(StatusCodes.Status500InternalServerError)]
        public async Task<IActionResult> AddNewAuditory(AddSubjectDTO dto)
        {
            try
            {
                await _service.AddNewSubjectAsync(dto);

                return Ok();
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
    }
}

```

```

[HttpGet]
[Route("getAllSubjects")]
[ProducesResponseType(typeof(SHORTSUBJECTDTO), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<IActionResult> GetAllSubjects()
{
    try
    {
        return Ok(await _service.GetAllSubjects());
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[HttpGet]
[Route("getFullSubjectInfoBySubjectId")]
[ProducesResponseType(typeof(SUBJECTFULLDTO), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<IActionResult> GetFullSubjectInfoBySubjectId([FromQuery] int subjectId)
{
    try
    {
        return Ok(await _service.GetFullSubjectInfoBySubjectId(subjectId));
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[HttpGet]
[Route("getFullSubjects")]
[ProducesResponseType(typeof(SUBJECTFULLDTO), StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<IActionResult> GetFullSubjects()
{
    try
    {
        return Ok(await _service.GetFullSubjects());
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
}
}

```

Б.9 Файл UserProfileController.cs

```

using BLL.Interfaces;
using CORE.Enums;

```

```

using Diploma_server.Hubs;
using Diploma_server.Infrastructure.Extensions;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.SignalR;
using Microsoft.EntityFrameworkCore;
using Repositories.DTOs.UserProfile;
using System.Globalization;

namespace Diploma_server.Controllers
{
    [ApiController]
    [ApiConventionType(typeof(DefaultApiConventions))]
    [Route("api/[controller]/")]
    [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
    public class UserProfileController : ControllerBase
    {
        private readonly IUserProfileService _service;

        public UserProfileController(IUserProfileService service)
        {
            _service = service;
        }

        [HttpGet]
        [Route("getCurrentUserProfile")]
        public async Task<IActionResult> GetUserProfile()
        {
            try
            {
                var currentUserId = User.GetUserIdOrDefault();
                var currentRoles = User.GetRoles().ToList();

                var user = await _service.GetUserProfileByIdAsync((int)currentUserId);
                user.Roles = currentRoles;

                return Ok(user);
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [HttpGet]
        [Route("getUserProfileByUserId")]
        public async Task<IActionResult> GetUserProfileByUserId([FromQuery]int userId)
        {
            try
            {
                return Ok(await _service.GetUserProfileByIdAsync(userId));
            }
            catch (Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }

        [HttpPost]
        [Route("getAllUsers")]
        public async Task<IActionResult> GetAllUsers([FromBody] BaseFilterUserDTO dto, [FromQuery] string culture)

```

```

{
    try
    {
        return Ok(await _service.GetAllUsersAsync(dto));
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[HttpPut]
[Route("changeUserProfile")]
public async Task<IActionResult> ChangeUserProfile([FromBody] UserProfileDTO dto)
{
    try
    {
        await _service.ChangeUserProfile(dto);

        return Ok();
    }
    catch(Exception ex)
    {
        return BadRequest(ex.Message);
    }
}

[HttpPut]
[Route("changeUserProfileSocialLinks")]
public async Task<IActionResult> ChangeUserProfileSocialLinks([FromBody] UserSocialLinksDTO dto)
{
    try
    {
        var currentUserId = User.GetUserIdOrDefault();

        await _service.ChangeUserProfileSocialLinks(dto, currentUserId.Value);

        return Ok();
    }
    catch(Exception e)
    {
        return BadRequest(e.Message);
    }
}

[HttpPut]
[Route("addGroupsToUser")]
public async Task<IActionResult> AddGroupsToUser([FromBody] AddGroupsDTO dto)
{
    try
    {
        await _service.AddGroupToUserAsync(dto);

        return Ok();
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}

[HttpPost]
[Route("addUsersAccesses")]

```

```

public async Task<IActionResult> AddGroupsAccesses([FromBody] SaveUserAccessesDTO dto)
{
    try
    {
        await _service.AddUsersAccesses(dto);

        return Ok();
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}

[HttpPost]
[Route("addUserRole")]
public async Task<IActionResult> AddUserRole([FromQuery] string role, [FromQuery] int userId)
{
    try
    {
        await _service.AddUserRole(role, userId);

        return Ok();
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}

[HttpPost]
[Route("addUsersRoles")]
public async Task<IActionResult> AddUsersRoles(AddUsersRolesDTO dto)
{
    try
    {
        await _service.AddUsersRoles(dto);

        return Ok();
    }
    catch (Exception e)
    {
        return BadRequest(e.Message);
    }
}
}
}

```

Б.10 Файл AuditoryRepository.cs

```

using DAL;
using DAL.Entities;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Repositories.DTOs.Auditories;
using Repositories.Interface;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Http.Headers;
using System.Text;

```

```

using System.Threading.Tasks;

namespace Repositories.Repositories
{
    public class AuditoryRepository : IAuditoryRepository
    {
        private readonly ApplicationContext _context;

        public AuditoryRepository(ApplicationContext context)
        {
            _context = context;
        }

        public async Task AddNewAuditoryAsync(AddAuditoryDTO dto)
        {
            if(dto.Corporus == null)
            {
                throw new Exception("CorpusCanNotBeEmpty");
            }

            if(dto.Name.IsNullOrEmpty())
            {
                throw new Exception("AuditoryNumberCanNotBeEmpty");
            }

            var checkExist = await _context.Auditories
                .Where(a => a.Name.ToLower() == dto.Name.ToLower())
                .FirstOrDefaultAsync();

            if(checkExist != null)
            {
                throw new Exception("ThatAudienceAlreadyExists");
            }

            var auditory = new Auditories
            {
                Corpus = dto.Corporus.Value,
                Name = dto.Name,
            };

            _context.Auditories.Add(auditory);

            await _context.SaveChangesAsync();
        }

        public async Task<List<AuditoryShortDTO>> GetAllAuditoriesAsync()
        {
            return await _context.Auditories
                .Select(ad => new AuditoryShortDTO
                {
                    Corpus = ad.Corporus,
                    Name = ad.Name,
                }).ToListAsync();
        }

        public async Task DeleteAuditoryAsync(AuditoryShortDTO dto)
        {
            var auditory = await _context.Auditories
                .FirstOrDefaultAsync(a => a.Name == dto.Name && a.Corporus == dto.Corporus);

            _context.Auditories.Remove(auditory);

            await _context.SaveChangesAsync();
        }
    }
}

```

```

    }
}
}

```

Б.11 Файл AuthorizationRepository.cs

```

using DAL.Entities;
using DAL;
using Microsoft.AspNetCore.Identity;
using Repositories.Interface;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;

namespace Repositories.Repositories
{
    public class AuthorizationRepository : IAuthorizationRepository
    {
        private readonly ApplicationDbContext _context;

        public AuthorizationRepository(ApplicationDbContext context, RoleManager<ApplicationRole> roleManager,
UserManager<ApplicationUser> userManager)
        {
            _context = context;
        }

        public async Task<UserProfile> GetUserProfileByAspNetUserIdAsync(string aspNetUserId)
        {
            return await _context.UserProfile.SingleOrDefaultAsync(u => u.AspNetUserId == aspNetUserId);
        }

        public async Task<UserProfile> SaveUserProfileAsync(UserProfile userProfile)
        {
            _context.UserProfile.Add(userProfile);
            await _context.SaveChangesAsync();

            return userProfile;
        }
    }
}

```

Б.11 Файл ChatRepository.cs

```

using CloudinaryAPI;
using CloudinaryDotNet.Actions;
using CORE.Enums;
using DAL;
using DAL.Entities;
using Microsoft.EntityFrameworkCore;
using Repositories.DTOs.Chat;
using Repositories.Interface;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

```

```
using System.Threading.Tasks;
```

```
namespace Repositories.Repositories
```

```
{
    public class ChatRepository : IChatRepository
    {
        private readonly ApplicationContext _context;

        public ChatRepository(ApplicationContext context)
        {
            _context = context;
        }

        public async Task<int> GetOrCreateNewSingleChatAndGetNewChatId(int userId, int chatUserId)
        {
            var chat = await _context.UserChats
                .Where(uc => (uc.UserId == userId || uc.UserId == chatUserId) && uc.Chat.Type == EChatType.Single)
                .ToListAsync();

            var findChat = chat.GroupBy(uc => uc.ChatId).ToList().Where(fc => fc.Count() > 1).FirstOrDefault();

            if(findChat != null)
            {
                return findChat.FirstOrDefault().ChatId;
            }

            else
            {
                var newChat = new Chat
                {
                    Name = "SingleChat",
                    Type = EChatType.Single
                };

                _context.Chats.Add(newChat);
                await _context.SaveChangesAsync();

                var userChat = new UserChat
                {
                    UserId = userId,
                    ChatId = newChat.ChatId
                };

                var inviterUserChat = new UserChat
                {
                    UserId = chatUserId,
                    ChatId = newChat.ChatId
                };

                _context.UserChats.Add(userChat);
                _context.UserChats.Add(inviterUserChat);

                await _context.SaveChangesAsync();

                return newChat.ChatId;
            }
        }

        public async Task<List<ChatsListDTO>> GetAllUsersCahtsAsync(int userId)
        {
            var chatsIds = await _context.Chats
```

```

.Include(c => c.UserChats)
.Where(c => c.UserChats.Any(uc => uc.UserId == userId) && c.Type == EChatType.Single)
.Select(uc => uc.ChatId)
.ToListAsync();

var userChats = await _context.UserChats
    .Where(uc => chatsIds.Contains(uc.ChatId) && uc.UserId != userId)
    .Select(uc => new ChatsListDTO
    {
        UserId = uc.UserId,
        ChatId = uc.ChatId,
        ChatType = uc.Chat.Type,
        Name = uc.User.Name,
        Surname = uc.User.Surname,
        UserChatId = uc.Id,
        UserName = uc.User.UserName,
        UserPhoto = uc.User.Photo != null ? CloudinaryHelper.GetImageUrl(uc.User.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg",
        LastMessage = uc.UserChatMessages.OrderByDescending(ucm => ucm.SendMessageDate).Select(ucm =>
new LastMessageDTO
    {
        MessageDate = ucm.SendMessageDate,
        MessageId = ucm.Id,
        MessageType = ucm.MessageType,
        Message = ucm.Mesage,
        SenderUserId = uc.UserId,
        SenderUserName = $"{uc.User.Name} {uc.User.Surname}",
        SenderUserPhoto = uc.User.Photo != null ? CloudinaryHelper.GetImageUrl(uc.User.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg"
    }).FirstOrDefault()
    })
    .ToListAsync();

return userChats;
}

public async Task<LastMessageDTO> SendNewMessage(SendMessageDTO dto, int userId)
{
    try
    {
        var user = await _context.UserProfile
            .SingleOrDefaultAsync(up => up.Id == userId);

        var userChatId = await _context.UserChats
            .AsNoTracking()
            .FirstOrDefaultAsync(uc => uc.UserId == userId && uc.ChatId == dto.ChatId);

        var message = new UserChatMessage
        {
            Message = dto.Message,
            MessageType = dto.MessageType,
            SendMessageDate = dto.SendMessageDate,
            UserChatId = userChatId.Id
        };

        _context.UserChatMessages.Add(message);

        await _context.SaveChangesAsync();

        return new LastMessageDTO
        {
            Message = message.Mesage,
            MessageDate = message.SendMessageDate,

```

```

        MessageId = message.Id,
        MessageType = message.MessageType,
        SenderUserId = userId,
        SenderUserName = $"{user.Name} {user.Surname}",
        SenderUserPhoto = user.Photo != null ? CloudinaryHelper.GetImageUrl(user.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg",
        ChatId = dto.ChatId
    };
}
catch (Exception ex)
{
    throw ex;
}
}

public async Task<List<LastMessageDTO>> GetAllChatMessagesAsync(int chatId)
{
    return await _context.UserChatMessages
        .Where(uc => uc.UserChat.ChatId == chatId)
        .OrderBy(uc => uc.SendMessageDate)
        .Select(ucm => new LastMessageDTO
        {
            Message = ucm.Mesage,
            MessageDate = ucm.SendMessageDate,
            MessageId = ucm.Id,
            MessageType = ucm.MessageType,
            SenderUserId = ucm.UserChat.UserId,
            SenderUserName = $"{ucm.UserChat.User.Name} {ucm.UserChat.User.Surname}",
            SenderUserPhoto = ucm.UserChat.User.Photo != null ?
CloudinaryHelper.GetImageUrl(ucm.UserChat.User.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg"
        }).ToListAsync();
}
}
}

```

Б.12 Файл GroupRepository.cs

```

using CloudinaryAPI;
using DAL;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Repositories.DTOs.Groups;
using Repositories.DTOs.UserProfile;
using Repositories.Interface;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Repositories.Repositories
{
    public class GroupRepository : IGroupRepository
    {
        private readonly ApplicationContext _context;

        public GroupRepository(ApplicationContext context)
        {
            _context = context;
        }
    }
}

```

```

    }

    public async Task<List<GroupsMainDTO>> GetAllGroupsAsync()
    {
        return await _context.Groups
            .Select(gr => new GroupsMainDTO
            {
                Id = gr.Id,
                Name = gr.Name,
                StudentsCount = gr.Students.Count
            }).ToListAsync();
    }

    public async Task AddNewGroupAsync(string? groupName)
    {
        if(groupName.IsNullOrEmpty())
        {
            throw new Exception("GroupNameCanNotBeEmpty");
        }

        if(groupName.Length > 0 && groupName.Length < 3)
        {
            throw new Exception("GroupNameCannotBeLessThanThreeLetter");
        }

        var existGroup = await _context.Groups
            .Where(g => g.Name.ToLower() == groupName.ToLower())
            .AnyAsync();

        if(existGroup)
        {
            throw new Exception("GroupIsAlreadyExist");
        }

        var group = new DAL.Entities.Group
        {
            Name = groupName,
            IsActive = true
        };

        _context.Groups.Add(group);

        await _context.SaveChangesAsync();
    }

    public async Task<GroupMainFullDTO> GetGroupsByIdAsync(int groupId)
    {
        return await _context.Groups
            .Where(gr => gr.Id == groupId)
            .Select(gr => new GroupMainFullDTO
            {
                Id = groupId,
                Name = gr.Name,
                Students = gr.Students.Select(st => new UserProfileShortDTO
                {
                    Id = st.Id,
                    Email = st.Email,
                    Name = st.Name,
                    Photo = st.Photo != null ? CloudinaryHelper.GetImageUrl(st.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg",
                    Surname = st.Surname,
                    UserName = st.UserName
                })
            })
    }

```

```

        }).ToList()
    }).FirstOrDefaultAsync();
}
}
}

```

Б.13 Файл ScheduleRepository.cs

```

using CORE.Enums;
using CORE.Helpers;
using DAL;
using DAL.Entities;
using Microsoft.EntityFrameworkCore;
using Repositories.DTOs.Auditories;
using Repositories.DTOs.Schedule;
using Repositories.DTOs.Subjects;
using Repositories.DTOs.UserProfile;
using Repositories.Extensions.IQueryable;
using Repositories.Interface;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Repositories.Repositories
{
    public class ScheduleRepository : IScheduleRepository
    {
        private readonly ApplicationContext _context;

        public ScheduleRepository(ApplicationContext context)
        {
            _context = context;
        }

        public async Task AddScheduleAsync(List<AddScheduleTableDTO> schedules, EWeekEnum day, int year, int term,
int groupId)
        {
            try
            {
                var newSchedules = new List<Schedule>();

                foreach(var schedule in schedules)
                {
                    var teacher = await _context.UserProfile
                        .AsNoTracking()
                        .FirstOrDefaultAsync(up => up.Id == schedule.Teacher.Id);

                    var auditory = await _context.Auditories
                        .AsNoTracking()
                        .FirstOrDefaultAsync(aud => aud.Name == schedule.Auditory.Name);

                    var subject = await _context.Subjects
                        .AsNoTracking()
                        .FirstOrDefaultAsync(s => s.Id == schedule.Subject.Id);

                    var group = await _context.Groups
                        .AsNoTracking()
                        .FirstOrDefaultAsync(g => g.Id == groupId);
                }
            }
        }
    }
}

```

```

        var newSchedule = new Schedule
        {
            Year = year,
            Semestr = term,
            GroupId = groupId,
            ScheduleWeekType = schedule.Type.Value,
            SubjectId = subject.Id,
            TeacherId = teacher.Id,
            AuditoryId = auditory.Name,
            DayOfWeek = day,
            PareName = schedule.PareCount
        };

        newSchedules.Add(newSchedule);
    }

    _context.AddRange(newSchedules);

    await _context.SaveChangesAsync();
}
catch(Exception ex)
{
    var a = 10;
}
}

public async Task<GetScheduleDTO> GetScheduleAsync(ScheduleFilterDTO filter)
{
    try
    {
        if(filter.CurrentUserId.HasValue)
        {
            filter.GroupId = await _context.UserProfile.Where(up => up.Id == filter.CurrentUserId)
                .Select(up => up.GroupId)
                .FirstOrDefaultAsync();
        }

        GetScheduleDTO model = await GetScheduleByFilterModel(filter);

        return model;
    }
    catch (Exception ex)
    {
        var a = 10;
        throw new Exception(ex.Message);
    }
}

private async Task<GetScheduleDTO> GetScheduleByFilterModel(ScheduleFilterDTO filter)
{
    var model = new GetScheduleDTO();
    var cultureId = LocalizationHelper.GetLangCodeByName(CultureInfo.CurrentUICulture.Name);

    model.Monday = await _context.Schedules
        .Filter(filter)
        .Where(sc => sc.DayOfWeek == EWeekEnum.Monday)
        .Select(sc => new ShowScheduleDTO
        {
            AuditoryName = sc.AuditoryId,
            DayOfWeek = sc.DayOfWeek,

```

```

        Id = sc.Id,
        TeacherId = sc.TeacherId,
        TeacherName = $"{sc.Teacher.Name} {sc.Teacher.Surname}",
        Type = sc.ScheduleWeekType,
        PareCount = sc.PareName,
        SubjectName = sc.Subject.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Any() ?
sc.Subject.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Select(sd => sd.Name).FirstOrDefault()
        : sc.Subject.SubjectDescriptions.Select(sd => sd.Name).FirstOrDefault()
    })
    .ToListAsync();

model.Tuesday = await _context.Schedules
    .Filter(filter)
    .Where(sc => sc.DayOfWeek == EWeekEnum.Tuesday)
    .Select(sc => new ShowScheduleDTO
    {
        AuditoryName = sc.AuditoryId,
        DayOfWeek = sc.DayOfWeek,
        Id = sc.Id,
        TeacherId = sc.TeacherId,
        PareCount = sc.PareName,
        TeacherName = $"{sc.Teacher.Name} {sc.Teacher.Surname}",
        Type = sc.ScheduleWeekType,
        SubjectName = sc.Subject.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Select(sd =>
sd.Name).FirstOrDefault()
        ?? sc.Subject.SubjectDescriptions.Select(sd => sd.Name).FirstOrDefault()
    })
    .ToListAsync();

model.Wednesday = await _context.Schedules
    .Filter(filter)
    .Where(sc => sc.DayOfWeek == EWeekEnum.Wednesday)
    .Select(sc => new ShowScheduleDTO
    {
        AuditoryName = sc.AuditoryId,
        DayOfWeek = sc.DayOfWeek,
        Id = sc.Id,
        PareCount = sc.PareName,
        TeacherId = sc.TeacherId,
        TeacherName = $"{sc.Teacher.Name} {sc.Teacher.Surname}",
        Type = sc.ScheduleWeekType,
        SubjectName = sc.Subject.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Select(sd =>
sd.Name).FirstOrDefault()
        ?? sc.Subject.SubjectDescriptions.Select(sd => sd.Name).FirstOrDefault()
    })
    .ToListAsync();

model.Thursday = await _context.Schedules
    .Filter(filter)
    .Where(sc => sc.DayOfWeek == EWeekEnum.Thursday)
    .Select(sc => new ShowScheduleDTO
    {
        AuditoryName = sc.AuditoryId,
        DayOfWeek = sc.DayOfWeek,
        PareCount = sc.PareName,
        Id = sc.Id,
        TeacherId = sc.TeacherId,
        TeacherName = $"{sc.Teacher.Name} {sc.Teacher.Surname}",
        Type = sc.ScheduleWeekType,
        SubjectName = sc.Subject.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Select(sd =>
sd.Name).FirstOrDefault()
        ?? sc.Subject.SubjectDescriptions.Select(sd => sd.Name).FirstOrDefault()
    })
    .ToListAsync();

```

```

        .ToListAsync();

model.Friday = await _context.Schedules
    .Filter(filter)
    .Where(sc => sc.DayOfWeek == EWeekEnum.Friday)
    .Select(sc => new ShowScheduleDTO
    {
        AuditoryName = sc.AuditoryId,
        DayOfWeek = sc.DayOfWeek,
        PareCount = sc.PareName,
        Id = sc.Id,
        TeacherId = sc.TeacherId,
        TeacherName = $"{sc.Teacher.Name} {sc.Teacher.Surname}",
        Type = sc.ScheduleWeekType,
        SubjectName = sc.Subject.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Select(sd =>
sd.Name).FirstOrDefault()
        ?? sc.Subject.SubjectDescriptions.Select(sd => sd.Name).FirstOrDefault()
    })
    .ToListAsync();

model.Saturday = await _context.Schedules
    .Filter(filter)
    .Where(sc => sc.DayOfWeek == EWeekEnum.Saturday)
    .Select(sc => new ShowScheduleDTO
    {
        AuditoryName = sc.AuditoryId,
        DayOfWeek = sc.DayOfWeek,
        Id = sc.Id,
        TeacherId = sc.TeacherId,
        PareCount = sc.PareName,
        TeacherName = $"{sc.Teacher.Name} {sc.Teacher.Surname}",
        Type = sc.ScheduleWeekType,
        SubjectName = sc.Subject.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Select(sd =>
sd.Name).FirstOrDefault()
        ?? sc.Subject.SubjectDescriptions.Select(sd => sd.Name).FirstOrDefault()
    })
    .ToListAsync();
return model;
    }
}
}

```

Б.14 Файл SubjectRepository.cs

```

using CloudinaryAPI;
using CORE.Helpers;
using DAL;
using DAL.Entities;
using DAL.Migrations;
using Microsoft.EntityFrameworkCore;
using Microsoft.IdentityModel.Tokens;
using Repositories.DTOs.Auditories;
using Repositories.DTOs.Subjects;
using Repositories.DTOs.UserProfile;
using Repositories.Interface;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;

```

```

using System.Threading.Tasks;

namespace Repositories.Repositories
{
    public class SubjectRepository : ISubjectRepository
    {
        private readonly ApplicationDbContext _context;

        public SubjectRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task AddNewSubjectAsync(AddSubjectDTO dto)
        {
            using var transaction = _context.Database.BeginTransaction();

            try
            {
                var subject = new Subject();

                _context.Subjects.Add(subject);

                await _context.SaveChangesAsync();

                var subjectDescriptions = new List<SubjectDescription>();

                if(!dto.UkrDescription.IsNullOrEmpty())
                {
                    subjectDescriptions.Add(new SubjectDescription
                    {
                        Description = dto.UkrDescription,
                        LanguageId = 1,
                        Name = dto.UkrName,
                        SubjectId = subject.Id
                    });
                }

                if (!dto.EngName.IsNullOrEmpty())
                {
                    subjectDescriptions.Add(new SubjectDescription
                    {
                        Description = dto.EngDescription,
                        LanguageId = 2,
                        Name = dto.EngName,
                        SubjectId = subject.Id
                    });
                }

                if(subjectDescriptions.Count > 0)
                {
                    _context.SubjectDescriptions.AddRange(subjectDescriptions);
                }

                var subAuditories = new List<SubjectAuditories>();
                var subTeachers = new List<SubjectTeachers>();

                foreach(var audit in dto.Auditories)
                {
                    subAuditories.Add(new SubjectAuditories
                    {
                        AuditorId = audit,

```

```

        SubjectId = subject.Id
    });
}

foreach (var teacher in dto.Teachers)
{
    subTeachers.Add(new SubjectTeachers
    {
        TeacherId = teacher,
        SubjectId = subject.Id
    });
}

_context.SubjectAuditories.AddRange(subAuditories);
_context.SubjectTeachers.AddRange(subTeachers);

await _context.SaveChangesAsync();

transaction.Commit();
}
catch (Exception ex)
{
    throw new Exception(ex.Message);
}
}

public async Task<List<ShortSubjectDTO>> GetAllSubjects()
{
    var cultureId = LocalizationHelper.GetLangCodeByName(CultureInfo.CurrentUICulture.Name);

    return await _context.Subjects
        .Select(sb => new ShortSubjectDTO
        {
            Id = sb.Id,
            Name = sb.SubjectDescriptions.FirstOrDefault(sd => sd.LanguageId == cultureId).Name ??
sb.SubjectDescriptions.FirstOrDefault().Name,
            Description = sb.SubjectDescriptions.FirstOrDefault(sd => sd.LanguageId == cultureId).Description ??
sb.SubjectDescriptions.FirstOrDefault().Description
        })
        .ToListAsync();
}

public async Task<SubjectFullOneDescDTO> GetFullSubjectInfoBySubjectId(int subjectId)
{
    var cultureId = LocalizationHelper.GetLangCodeByName(CultureInfo.CurrentUICulture.Name);

    return await _context.Subjects
        .AsSplitQuery()
        .Where(sb => sb.Id == subjectId)
        .Select(sb => new SubjectFullOneDescDTO
        {
            Id = sb.Id,
            Auditories = sb.SubjectAuditories.Select(sa => new AuditoryShortDTO
            {
                Corpus = sa.Auditory.Corpus,
                Name = sa.Auditory.Name
            }).ToList(),
            Description = sb.SubjectDescriptions.Where(desc => desc.LanguageId == cultureId).Select(sd => new
SubjectDescriptionDTO
            {
                Description = sd.Description,
                Id = sd.Id,
                Name = sd.Name,
                LanguageId = sd.LanguageId,

```

```

        SubjectId = sd.SubjectId
    }).FirstOrDefault()
    ?? sb.SubjectDescriptions.Select(sd => new SubjectDescriptionDTO
    {
        Description = sd.Description,
        Id = sd.Id,
        Name = sd.Name,
        LanguageId = sd.LanguageId,
        SubjectId = sd.SubjectId
    }).FirstOrDefault(),
    Teachers = sb.SubjectTeachers.Select(st => new UserProfileShortDTO
    {
        Name = st.Teacher.Name,
        Email = st.Teacher.Email,
        Id = st.Teacher.Id,
        Photo = st.Teacher.Photo != null ? CloudinaryHelper.GetImageUrl(st.Teacher.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg",
        Surname = st.Teacher.Surname,
        UserName = st.Teacher.UserName
    }).ToList()
    }).FirstOrDefaultAsync();
}

public async Task<List<SubjectFullOneDescDTO>> GetFullSubjects()
{
    var cultureId = LocalizationHelper.GetLangCodeByName(CultureInfo.CurrentUICulture.Name);

    return await _context.Subjects
        .AsSplitQuery()
        .Select(sb => new SubjectFullOneDescDTO
        {
            Id = sb.Id,
            Auditories = sb.SubjectAuditories.Select(sa => new AuditoryShortDTO
            {
                Corpus = sa.Auditory.Corpus,
                Name = sa.Auditory.Name
            }).ToList(),
            Description = sb.SubjectDescriptions.Where(sd => sd.LanguageId == cultureId).Select(sd => new
SubjectDescriptionDTO
            {
                Description = sd.Description,
                Id = sd.Id,
                Name = sd.Name,
                LanguageId = sd.LanguageId,
                SubjectId = sd.SubjectId
            }).FirstOrDefault(),
            Teachers = sb.SubjectTeachers.Select(st => new UserProfileShortDTO
            {
                Name = $"{st.Teacher.Name} {st.Teacher.Surname}",
                Email = st.Teacher.Email,
                Id = st.Teacher.Id,
                Photo = st.Teacher.Photo != null ? CloudinaryHelper.GetImageUrl(st.Teacher.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg",
                Surname = st.Teacher.Surname,
                UserName = st.Teacher.UserName
            }).ToList()
        }).ToListAsync();
    }
}
}

```

Б.15 Файл UserProfileRepository.cs

```

using CloudinaryAPI;
using CORE.Enums;
using DAL;
using DAL.Entities;
using Microsoft.EntityFrameworkCore;
using Repositories.DTOs.UserProfile;
using Repositories.Extensions.IQueryable;
using Repositories.Interface;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Repositories.Repositories
{
    public class UserProfileRepository : IUserProfileRepository
    {
        private readonly ApplicationContext _context;

        public UserProfileRepository(ApplicationContext context)
        {
            _context = context;
        }

        public async Task<UserProfileDTO> GetUserProfileByIdAsync(int UserProfileId)
        {
            return await _context.UserProfile.Where(up => up.Id == UserProfileId)
                .Select(up => new UserProfileDTO
                {
                    Id = UserProfileId,
                    Name = up.Name,
                    AspNetUserId = up.AspNetUserId,
                    BirthDate = up.BirthDate,
                    Email = up.Email,
                    Photo = up.Photo != null ? CloudinaryHelper.GetImageUrl(up.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg",
                    RegistrationTime = up.RegistrationTime,
                    Surname = up.Surname,
                    UserName = up.UserName,
                    FacebookLink = up.FacebookLink,
                    InstagramLink = up.InstagramLink,
                    LinkedInLink = up.LinkedinLink,
                    TwitchLink = up.TwitchLink,
                    TwitterLink = up.TwitterLink,
                    YoutubeLink = up.YoutubeLink,
                    UserAccesses = up.UserAccesses.Select(ua => ua.UserAcces).ToList()
                }).SingleOrDefaultAsync();
        }

        public async Task<List<UserProfileDTO>> GetAllUsersAsync(BaseFilterUserDTO dto)
        {
            return await _context.UserProfile
                .Filter(dto)
                .Select(up => new UserProfileDTO
                {
                    Id = up.Id,
                    Name = up.Name,
                    AspNetUserId = up.AspNetUserId,

```

```

        BirthDate = up.BirthDate,
        Email = up.Email,
        Photo = up.Photo != null ? CloudinaryHelper.GetImageUrl(up.Photo) :
"https://res.cloudinary.com/dbsmzvvd8/image/upload/v1699802223/diploma/za6zifaregpuf4lyi659.svg",
        RegistrationTime = up.RegistrationTime,
        Surname = up.Surname,
        UserName = up.UserName
    }).ToListAsync();
}

public async Task ChangeUserProfile(UserProfileDTO userProfile)
{
    var user = await _context.UserProfile
        .SingleOrDefaultAsync(up => up.Id == userProfile.Id);

    var emailUser = await _context.UserProfile
        .FirstOrDefaultAsync(up => up.Id != userProfile.Id && up.UserName.ToLower() ==
userProfile.UserName.ToLower());

    if(emailUser != null)
    {
        throw new Exception("UserWitSameUserNameIsAlreadyExist");
    }

    user.Name = userProfile.Name;
    user.Surname = userProfile.Surname;
    user.Email = userProfile.Email;
    user.UserName = userProfile.UserName;
    user.BirthDate = userProfile.BirthDate;

    if(userProfile.Photo != null)
    {
        user.Photo = userProfile.Photo;
    }

    await _context.SaveChangesAsync();
}

public async Task ChangeUserProfileSocialLinks(UserSocialLinksDTO dto, int userId)
{
    var user = await _context.UserProfile
        .SingleOrDefaultAsync(up => up.Id == userId);

    user.FacebookLink = dto.FacebookLink;
    user.LinkedinLink = dto.LinkedinLink;
    user.InstagramLink = dto.InstagramLink;
    user.YoutubeLink = dto.YoutubeLink;
    user.TwitchLink = dto.TwitchLink;
    user.TwitterLink = dto.TwitterLink;

    await _context.SaveChangesAsync();
}

public async Task AddGroupToUserAsync(AddGroupsDTO dto)
{
    var oldUsers = await _context.UserProfile
        .Where(up => up.GroupId == dto.GroupId)
        .ToListAsync();

    var users = await _context.UserProfile
        .Where(up => dto.UserIds.Any(ui => ui == up.Id))
        .ToListAsync();
}

```

```

foreach(var oldUser in oldUsers)
{
    oldUser.GroupId = null;
}

foreach (var user in users)
{
    user.GroupId = dto.GroupId;
}

await _context.SaveChangesAsync();
}

public async Task AddUsersAccesses(SaveUserAccessesDTO dto)
{
    var userAccesses = await _context.UserAccesses
        .Where(ua => ua.UserId == dto.UserId).ToListAsync();

    _context.UserAccesses.RemoveRange(userAccesses);

    await _context.SaveChangesAsync();

    var accesses = new List<UserAccesses>();

    foreach(var access in dto.UserAccesses)
    {
        accesses.Add(new UserAccesses
        {
            UserAcces = access,
            UserId = dto.UserId
        });
    }

    _context.UserAccesses.AddRange(accesses);

    await _context.SaveChangesAsync();
}

public async Task AddUserRole(string roleName, int userId)
{
    var userAppId = await _context.UserProfile
        .Where(up => up.Id == userId)
        .Select(up => up.ApplicationUser.Id)
        .FirstOrDefaultAsync();

    var role = await _context.Roles
        .Where(r => r.Name.ToLower() == roleName.ToLower())
        .Select(r => r.Id)
        .FirstOrDefaultAsync();

    var newUserRole = new ApplicationUserRole
    {
        RoleId = role,
        UserId = userAppId
    };

    _context.UserRoles.Add(newUserRole);

    await _context.SaveChangesAsync();
}

public async Task AddUsersRoles(AddUsersRolesDTO dto)
{

```

```

var userAppId = await _context.UserProfile
    .Where(up => dto.UsersIds.Any(ui => up.Id == ui))
    .Select(up => up.ApplicationUser.Id)
    .ToListAsync();

var role = await _context.Roles
    .Where(r => r.Name.ToLower() == dto.Role.ToLower())
    .Select(r => r.Id)
    .FirstOrDefaultAsync();

var usersRoles = await _context.UserRoles
    .Where(r => r.RoleId == role)
    .ToListAsync();

_context.RemoveRange(usersRoles);

await _context.SaveChangesAsync();

foreach(var userId in userAppId)
{
    var newUserRole = new ApplicationUserRole
    {
        RoleId = role,
        UserId = userId
    };

    _context.UserRoles.Add(newUserRole);
}

await _context.SaveChangesAsync();
}
}
}

```

Б.16 Файл Chat.cs

```

using BLL.Interfaces;
using CORE.Enums;
using Diploma_server.Infrastructure.Extensions;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.SignalR;
using Repositories.DTOs.Chat;
using System.Collections.Concurrent;
using System.Diagnostics;
using System.Linq;
using static Microsoft.EntityFrameworkCore.DbLoggerCategory.Database;

namespace Diploma_server.Hubs
{
    public static class UserHandler
    {
        public static Dictionary<string, int?> ConnectedIds = new Dictionary<string, int?>();
        public static Dictionary<string, DateTime> _connections = new Dictionary<string, DateTime>();
    }
    [Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
    public class Chat : Hub
    {
        private readonly IHubContext<Chat> _hubContext;
        private readonly IChatService _chatService;
    }
}

```

```

public Chat(IHubContext<Chat> hubContext, IChatService chatService)
{
    _hubContext = hubContext;
    _chatService = chatService;
}

public async Task Ping()
{
    if(UserHandler._connections.Keys.Any(k => k == Context.ConnectionId))
    {
        UserHandler._connections[Context.ConnectionId] = DateTime.UtcNow;
        if(UserHandler.ConnectedIds.Keys.Any(k => k != Context.ConnectionId))
        {
            UserHandler.ConnectedIds.Add(Context.ConnectionId, Context.User.GetUserIdOrDefault());
        }
    }
    else
    {
        UserHandler._connections.Add(Context.ConnectionId, DateTime.UtcNow);
        UserHandler.ConnectedIds.Add(Context.ConnectionId, Context.User.GetUserIdOrDefault());
    }
    await Clients.Caller.SendAsync("Pong");
}

public void CheckConnections()
{
    var now = DateTime.UtcNow;
    var disconnectedConnections = UserHandler._connections
        .Where(pair => (now - pair.Value).TotalSeconds > 30)
        .Select(pair => pair.Key)
        .ToList();

    foreach (var connectionId in disconnectedConnections)
    {
        UserHandler._connections.Remove(connectionId);
        UserHandler.ConnectedIds.Remove(connectionId);
        Clients.All.SendAsync("Online", UserHandler.ConnectedIds.Values);
    }
}

public override async Task OnConnectedAsync()
{
    var userId = Context.User.GetUserIdOrDefault();
    UserHandler._connections.TryAdd(Context.ConnectionId, DateTime.UtcNow);
    Console.WriteLine("ChatHub Connected!");

    UserHandler.ConnectedIds.Add(Context.ConnectionId, Context.User.GetUserIdOrDefault());

    //GetAllUsersCahtsAsync
    var chats = await _chatService.GetAllUsersCahtsAsync((int)userId);

    foreach (var chat in chats)
    {
        await Groups.AddToGroupAsync(Context.ConnectionId, chat.ChatId.ToString());
    }

    await Clients.All.SendAsync("Online", UserHandler.ConnectedIds.Values);
    await base.OnConnectedAsync();
}

public async Task Send(string nick, string message)
{
    await Clients.All.SendAsync("Send", nick, message);
}

```

```

    }

    public override async Task OnDisconnectedAsync(Exception? exception)
    {
        UserHandler._connections.Remove(Context.ConnectionId);
        Trace.WriteLine(Context.ConnectionId + " - disconnected");
        UserHandler.ConnectedIds.Remove(Context.ConnectionId);

        await Clients.All.SendAsync("Online", UserHandler.ConnectedIds.Values);

        await base.OnDisconnectedAsync(exception);
    }

    public async Task Online(string nick, EChatEvent chatEvent)
    {
        await Clients.All.SendAsync("Online", UserHandler.ConnectedIds.Values);
    }

    public async Task ChangeUserData()
    {
        await Clients.All.SendAsync("ChangeUserData");
    }

    public async Task SendNewMessage(SendMessageDTO dto)
    {
        var userId = Context.User.GetUserIdOrDefault();
        var message = await _chatService.SendNewMessage(dto, (int)userId);

        Clients.Group(dto.ChatId.ToString()).SendAsync("GetNewMessage", message);
        Clients.GroupExcept(dto.ChatId.ToString(), Context.ConnectionId).SendAsync("GetNewMessageNotification",
message.SenderUserName);
    }
}
}

```

Б.17 Файл en.json

```

{
    "Login": "Login | Log in",
    "NUZP": "NUZP",
    "YourEmail": "Your Email",
    "YourPassword": "Your Password",
    "IfYouDoNotHaveAnAccountYetPlease": "If you do not have an account yet, please",
    "IfYouAlreadyHaveAnAccountPleaseFeelFreeTo": "If you already have an account, please",
    "Register": "Register",
    "It": "It",
    "YourName": "Your name",
    "YourSurname": "Your surname",
    "YourBirthday": "Your birthday",
    "ToIt": "To it",
    "ItemIsRequired": "Item is required",
    "EmailIsRequired": "E-mail is required",
    "EmailMustBeValid": "E-mail must be valid",

```

"MinimumSixCharacters": "Minimum Six Characters",
 "RepeatPassword": "Repeat password",
 "PasswordRequired": "Password Required",
 "PasswordsDoesNotMatch": "Passwords does not match",
 "ThePasswordCannotBeLessThanSixCharacters": "The password cannot be less than six characters",
 "Profile": "Profile",
 "Chat": "Chat",
 "Logout": "Logout",
 "Language": "Language",
 "Theme": "Theme",
 "Schedule": "Schedule",
 "Marks": "Marks",
 "Info": "Info",
 "Teachers": "Teachers",
 "Groups": "Groups",
 "InformationPage": "Information page",
 "InfoMain": "This page was created to help students use the NUZP-system website. In this section, we will tell you how to use the site, help you locate university buildings, and inform you about some of the details of this project.",
 "HowToGetToTheFirstBuildingOfTheUniversity": "How to get to the {buildingNumber} building of the university?",
 "First": "First",
 "Second": "Second",
 "Third": "Third",
 "Fourth": "Fourth",
 "Fifth": "Fifth",
 "Sporty": "Sporty",
 "ItsSimpleJustFollowTheMapBelow": "it's simple, just follow the map below.",
 "WhatIsTheTabFor": "What is the {tabName} tab for?",
 "StudentProfileReport": "Student profile report!",
 "WelcomeToYourStudentProfileOnOurWebsite": "Welcome to your student profile on our website! Your profile will help other users to learn more about you and your achievements.",
 "YourProfileIsABusinessCardThatHelpsOther": "Your profile is a business card that helps other users get to know you better. Don't forget to save your changes to keep your profile up to date. Thank you for being with us!",
 "TheScheduleTabIsNeeded": "The schedule tab is needed so that a student can monitor the schedule for his/her group.",
 "TheChatTabIsNecessaryFor": "The chat tab is necessary for students to communicate with other students as well as with professors.",
 "TheMarksTabIsNeeded": "The marks tab is needed so that the student can view their grades.",
 "TheTeachersTabIsNecessary": "The teachers tab is necessary for the student to be able to view all teachers, their schedule, as well as the teacher's free time."

"AreYouSureYouWantToLogOutOfTheAccount": "Are you sure you want to log out of the account?",
 "Yes": "Yes",
 "No": "No",
 "YourUserName": "Your username",
 "Users": "Users",
 "Photo": "Photo",
 "IncorrectLoginOrPassword": "Incorrect login or password",
 "SomethingWentWrongWhileUploadingThePhoto": "Something went wrong while uploading the photo",
 "SomethingWentWrongWhileUploadingNewUserData": "Something went wrong while uploading new user data",
 "Socials": "Socials",
 "Save": "Save",
 "Message": "Message",
 "YouHaveGotANewMessageFrom": "You have got a new message from",
 "AddGroup": "Add group",
 "EnterGroupName": "Enter group name",
 "Students": "Students",
 "AddStudents": "Add students",
 "Accesses": "Accesses",
 "Private": "Private",
 "IsPhoneShow": "Is your phone show to other users",
 "IsMarksShow": "Is your marks show to other users",
 "IsBirthShow": "Is your birth show to other users",
 "IsFacebookShow": "Is your Facebook show to other users",
 "IsLinkedInShow": "Is your LinkedIn show to other users",
 "IsInstagramShow": "Is your Instagram show to other users",
 "IsYoutubeShow": "Is your Youtube show to other users",
 "IsTwitchShow": "Is your Twitch show to other users",
 "IsTwitterShow": "Is your Twitter show to other users",
 "IsGroupShow": "Is your Groups show to other users",
 "IsScheduleShow": "Is your Schedule show to other users",
 "Name": "Name",
 "Surname": "Surname",
 "Birthday": "Birthday",
 "Email": "Email",
 "UserName": "Username",
 "RegistrationTime": "Registration time",
 "AddTeacher": "Add teacher",
 "Subjects": "Subjects",
 "Auditories": "Auditories",
 "CreateSchedule": "Create schedule",
 "MakeReplacement": "Make replacement",

"ShowSchedule": "Show schedule",
 "AddAuditory": "Add audience",
 "AuditoryName": "audience name",
 "AuditoryCorps": "audience corps",
 "UserWitSameUserNameIsAlreadyExist": "User with same userName is already exist",
 "GroupIsAlreadyExist": "Group is already exist",
 "GroupNameCanNotBeEmpty": "Group name can not be empty",
 "GroupNameCannotBeLessThenThreeLetter": "Group name can not be less then three letter",
 "DuplicateUserName": "User with the same email are already exist in the system",

"MainTitle": "Our software is an advanced tool for effective monitoring of academic performance and controlling changes in the learning process. Thanks to a number of innovative features, we provide full control over all aspects of education. The user authorisation system guarantees security and limited access. Our users can create chats for easy communication and information exchange. The ability to view users by group makes managing group processes as easy as possible. To optimise the learning process, we offer a wide range of features, such as searching for a group schedule by number, sorting groups by administrator, and viewing user grades. Keep track of schedules and add grades by the teacher, providing clear information about academic achievements. We care about convenience and personalisation. Users can create their own group schedules, add or hide information of their choice. The notification system immediately notifies users of any important changes, and the ability to send text messages simplifies communication between all participants. Special attention is paid to localisation. The application interface is developed in both English and Ukrainian to ensure maximum comfort for users from different countries and language environments. Our software is not just a tool, but a key to optimising and improving learning processes.",

"CorpusCanNotBeEpmpty": "Corpus can not be epmpty",
 "AuditoryNumberCanNotBeEpmpty": "The audience cannot be empty",
 "AuditoryHasBeenAdded": "Audience has been added",
 "ThatAudienceAlreadyExists": "That audience already exists",
 "AddAudience": "Add audience",
 "SubjectName": "Subject name",
 "SubjectDescription": "Subject description",
 "Audiences": "Audiences",
 "AddSubject": "Add subject",
 "Year": "Year",
 "Semester": "Semester",
 "Monday": "Monday",
 "Tuesday": "Tuesday",
 "Wednesday": "Wednesday",
 "Thursday": "Thursday",
 "Friday": "Friday",
 "Saturday": "Saturday",
 "Subject": "Subject",
 "Enter": "Login",
 "Registration": "Registration",

```

"NUZPSocial": "NUZP Social",
"Chiselnik": "Numerator",
"Znamenik": "Denominator",
"Pare": "Pare",
"Teacher": "Teacher",
"Auditory": "Audience",
"Type": "Type",
"TermShouldBeOnlyOneOrTwo": "Semestr should be only 1 or 2",
"MinYearShouldBeMoreThen2000": "Min year should be more then 2000",
"MaxYearIsShuoldBeLessThenCurrentYearPlus1": "Max year shuold be less then current year plus 1",
"Description": "Description",
"CannotSaveSubjectWithoutAnySubjectsName": "Cannot save subject without any subjects name",
"PleaseSelectAnyTeachers": "Please select any teachers",
"PleaseSelectAnyAuditories": "Please select any auditories"
}

```

Б.18 Файл uk.json

```

{
  "Login": "Увійти | Увійдіть",
  "NUZP": "НУЗП",
  "YourEmail": "Ваш емейл",
  "YourPassword": "Ваш пароль",
  "IfYouDoNotHaveAnAccountYetPlease": "Якщо у вас ще немає облікового запису, будь ласка",
  "IfYouAlreadyHaveAnAccountPleaseFeelFreeTo": "Якщо ви вже маєте обліковий запис, будь ласка",
  "Register": "Зареєструйте | Зареєструватися",
  "It": "Його | нього",
  "In": "до | в",
  "YourName": "Ваше Ім'я",
  "YourSurname": "Ваше прізвище",
  "YourBirthday": "Твій день народження",
  "ToIt": "До нього",
  "ItemIsRequired": "Обов'язковий пункт",
  "EmailIsRequired": "Електронна пошта обов'язкова",
  "EmailMustBeValid": "Неправильний формат адреси електронної пошти",
  "MinimumSixCharacters": "Мінімум шість символів",
  "RepeatPassword": "Повторний пароль",
  "PasswordRequired": "Необхідно ввести пароль",
  "PasswordsDoesNotMatch": "Паролі не співпадають",
  "ThePasswordCannotBeLessThanSixCharacters": "Пароль не може бути менше ніж шість символів",
  "Profile": "Профіль",
  "Chat": "Чат",

```

"Logout": "Вийти",

"Language": "Мова",

"Theme": "Тема",

"Schedule": "Розклад",

"Marks": "Оцінки",

"Info": "Інфо",

"Teachers": "Викладачі",

"Groups": "Групи",

"InformationPage": "Інформаційна сторінка",

"InfoMain": "Ця сторінка створена для допомоги студентам у використанні сайту 'НУЗП-система'. У цьому розділі ми розповімо Вам, як користуватися сайтом, допоможемо Вам у розташуванні корпусів університету, а також проінформуємо Вас про деякі дрібниці цього проєкту.",

"HowToGetToTheFirstBuildingOfTheUniversity": "Як дістатися до {buildingNumber} корпусу університету?",

"First": "Перший | Першого",

"Second": "Другий | Другого",

"Third": "Третій | Третього",

"Fourth": "Четвертий | Четвертого",

"Fifth": "П'ятий | П'ятого",

"Sporty": "Спортивний | Спортивного",

"ItsSimpleJustFollowTheMapBelow": "Усе просто, просто слідуйте за картою, яка розташована знизу.",

"WhatIsTheTabFor": "Для чого потрібна вкладка {tabName}?",

"StudentProfileReport": "Звіт профілю студента!",

"WelcomeToYourStudentProfileOnOurWebsite": "Ласкаво просимо до вашого профілю студента на нашому сайті! Ваш профіль допоможе іншим користувачам дізнатися більше про вас та ваші досягнення.",

"YourProfileIsABusinessCardThatHelpsOther": "Ваш профіль - це візитівка, яка допоможе іншим користувачам краще пізнати вас. Не забудьте зберегти зміни, щоб ваш профіль був завжди актуальним. Дякуємо, що ви з нами!",

"TheScheduleTabIsNeeded": "Вкладка розкладу потрібна щоб студент міг моніторити розклад по своїй групі.",

"TheChatTabIsNecessaryFor": "Вкладка чат потрібна для того, щоб студент спілкувався з іншими студентами, а так само з викладачами.",

"TheMarksTabIsNeeded": "Вкладка оцінок потрібна щоб студент міг переглядати свої оцінки.",

"TheTeachersTabIsNecessary": "Вкладка викладачі потрібно щоб студент міг переглядати, усіх вчителів, їх графік, а також вільний час викладача.",

"AreYouSureYouWantToLogOutOfTheAccount": "Ви впевненні що хочете вийти з акаунта?",

"Yes": "Так",

"No": "Ні",

"YourUserName": "Ваш юзернейм",

"Users": "Користувачі",

"Photo": "Фото",

"IncorrectLoginOrPassword": "Неправильний логін чи пароль",

"SomethingWentWrongWhileUploadingThePhoto": "Щось пішло не так під час завантаження фотографії",

"SomethingWentWrongWhileUploadingNewUserData": "Щось пішло не так під час завантаження нових даних користувача",

"Socials": "Соціальні мережі",

"Save": "Зберегти",

"Message": "Повідомлення",

"YouHaveGotANewMessageFrom": "У вас нове повідомлення від",

"AddGroup": "Додати групу",

"EnterGroupName": "Впишіть номер групи",

"Students": "Студенти",

"AddStudents": "Додати студентів",

"Accesses": "Доступи",

"Private": "Приватність",

"IsPhoneShow": "Чи показується ваш телефон іншим користувачам",

"IsMarksShow": "Чи показується ваші позначки іншим користувачам",

"IsBirthShow": "Чи показується ваше народження іншим користувачам",

"IsFacebookShow": "Чи показується ваш Facebook іншим користувачам",

"IsLinkedInShow": "Чи показується ваш LinkedIn іншим користувачам",

"IsInstagramShow": "Чи показується ваш Instagram іншим користувачам",

"IsYoutubeShow": "Чи показується ваш Youtube іншим користувачам",

"IsTwitchShow": "Чи показується ваше Twitch іншим користувачам",

"IsTwitterShow": "Чи показується ваш Twitter іншим користувачам",

"IsGroupShow": "Чи показуються ваші групи іншим користувачам",

"IsScheduleShow": "Чи показується ваш розклад іншим користувачам",

"Name": "Ім'я",

"Surname": "Прізвище",

"Birthday": "Дата народження",

"Email": "Електронна пошта",

"UserName": "Ім'я користувача",

"RegistrationTime": "Час реєстрації",

"AddTeacher": "Додати викладача",

"Subjects": "Предмети",

"Auditories": "Аудиторії",

"CreateSchedule": "Створити розклад",

"MakeReplacement": "Створити заміну",

"ShowSchedule": "Перегляд розкладу",

"AddAuditory": "Додати аудиторію",

"AuditoryName": "Назва аудиторії",

"AuditoryCorps": "Корпус аудиторії",

"DuplicateUserName": "Користувач із такою електронною поштою вже існує в системі",

"UserWithSameUserNameAlreadyExist": "Користувач з таким же юзернеймом вже існує",

"GroupAlreadyExist": "Така група вже існує",

"GroupNameCannotBeEmpty": "Ім'я групи не може бути порожнім",

"GroupNameCannotBeLessThanThreeLetter": "Назва групи не може складатися менше ніж з трьох літер",

"MainTitle": "Наше програмне забезпечення є вдосконалим інструментом для здійснення ефективного моніторингу успішності та контролю змін у навчальному процесі. Завдяки ряду інноваційних функцій, ми забезпечуємо повний контроль над усіма аспектами освіти. Система авторизації користувачів гарантує безпеку та обмежений доступ. Наші користувачі можуть створювати чати для зручного спілкування та обміну інформацією. Можливість перегляду користувачів за групами робить управління груповими процесами максимально простим. Для оптимізації навчального процесу ми пропонуємо широкий спектр можливостей, таких як пошук розкладу групи за номером, сортування груп адміністратором та перегляд оцінок користувача. Відстежуйте розклади та додавайте оцінки викладачем, забезпечуючи зрозумілу інформацію про навчальні досягнення. Ми дбаємо про зручність та персоналізацію. Користувачі можуть створювати власні розклади по групах, додавати чи приховувати інформацію за власним вибором. Система сповіщень негайно повідомляє користувачів про будь-які важливі зміни, а можливість відправлення текстових повідомлень спрощує комунікацію між усіма учасниками. Особлива увага приділяється локалізації. Інтерфейс застосунку розроблений як англійською, так і українською мовами, щоб забезпечити максимальний комфорт користувачів з різних країн та мовних середовищ. Наше програмне забезпечення - це не просто інструмент, але ключ до оптимізації та покращення процесів навчання.",

"CorpusCannotBeEmpty": "Корпус не може бути порожнім",

"AuditoryNumberCannotBeEmpty": "Аудиторія не може бути порожньою",

"AuditoryHasBeenAdded": "Аудиторію додано",

"ThatAudienceAlreadyExists": "Така аудиторія вже існує",

"AddAudience": "Додати аудиторію",

"SubjectName": "Назва предмету",

"SubjectDescription": "Опис предмету",

"Audiences": "Аудиторії",

"AddSubject": "Додати предмет",

"Year": "Рік",

"Semester": "Семестр",

"Monday": "Понеділок",

"Tuesday": "Вівторок",

"Wednesday": "Середа",

"Thursday": "Четвер",

"Friday": "П'ятниця",

"Saturday": "Суббота",

"Subject": "Предмет",

"Enter": "Вхід",

"Registration": "Реєстрація",

"NUZPSocial": "НУЗП Social",

```

    "Chiselnik": "Чисельник",
    "Znamenik": "Знаменник",
    "Pare": "Пара",
    "Teacher": "Викладач",
    "Auditory": "Аудиторія",
    "Type": "Тип",
    "TermShouldBeOnlyOneOrTwo": "Семестр має бути лише 1 або 2",
    "MinYearShouldBeMoreThen2000": "Мінімальний рік повинен бути більше 2000",
    "MaxYearIsShuoldBeLessThenCurrentYearPlus1": "Максимальний рік має бути меншим за поточний
    плюс 1",
    "Description": "Опис",
    "CannotSaveSubjectWithoutAnySubjectsName": "Неможливо зберегти предмет без назви",
    "PleaseSelectAnyTeachers": "Будь ласка, оберіть будь-якого викладача",
    "PleaseSelectAnyAuditories": "Будь ласка, оберіть будь-яку аудиторію"
  }
}

```

Б.19 Файл index.vue

```

<template>
  <v-row no-gutters justify="center" align="center">
    <v-col cols="12" sm="8" md="6">
      <v-card>
        <v-card-title class="headline">
          NUZP Social
        </v-card-title>
        <v-card-text class="text-body-1">
          {{ $t('MainTitle') }}
        </v-card-text>
        <v-card-actions>
          <v-spacer />
        </v-card-actions>
      </v-card>
    </v-col>
  </v-row>
</template>

<script lang="ts">
import { Vue, Component } from "nuxt-property-decorator";

@Component({
  name: 'IndexPage',
})

```

```
export default class InexBlock extends Vue {

  private head() {
    return {
      title: this.$t('NUZPSocial')
    }
  }

}

</script>
```

Б.20 Файл login.vue

```
<template>
  <v-container class="fill-height" fluid>
    <v-row justify="center" no-gutters>
      <v-col cols="12">
        <v-row justify="center" no-gutters>
          <v-col class="login-block" cols="12">
            <login-main-block />
          </v-col>
        </v-row>
      </v-col>
    </v-row>
  </v-container>
</template>

<script lang="ts">
import { Component, Vue } from 'nuxt-property-decorator';

import LoginMainBlock from '~/components/login/login-main-block.vue';

@Component({
  components: {
    LoginMainBlock
  },
  layout: 'default'
})
export default class Login extends Vue {
  private head() {
```

```

    return {
      title: this.$t('Enter')
    }
  }
}
</script>

```

```

<style scoped>
  .login-block {
    max-width: 400px;
  }
</style>

```

Б.21 Файл register.vue

```

<template>
  <v-container class="fill-height" fluid>
    <v-row justify="center" no-gutters>
      <v-col cols="12">
        <v-row justify="center" no-gutters>
          <v-col class="login-block" cols="12">
            <register-main-block />
          </v-col>
        </v-row>
      </v-col>
    </v-row>
  </v-container>
</template>

<script lang='ts'>
import Vue from 'vue';
import { Component } from 'nuxt-property-decorator';

import RegisterMainBlock from '~/components/register/register-main-block.vue';

@Component({
  components: {
    RegisterMainBlock
  },

```

```

        layout: 'default'
    })
    export default class Login extends Vue {

        private head() {
            return {
                title: this.$t('Registration')
            }
        }
    }
</script>

<style scoped>
    .login-block {
        max-width: 400px;
    }
</style>

```

Б.22 Файл `auditory-service.ts`

```

import { AddAuditoryModel } from "~/models/auditories/AddAuditoryModel";
import { $axios } from "~/utils/api";

const api = 'api/auditory';

class AuditoryService {

    public async addNewAuditory(dto: AddAuditoryModel): Promise<void> {
        return $axios.$post<void>(`${api}/addNewAuditory`, dto);
    }

    public async getAllAuditories(): Promise<Array<AddAuditoryModel>> {
        return $axios.$get<Array<AddAuditoryModel>>(`${api}/getAllAuditories`);
    }

    public async removeAuditory(dto: AddAuditoryModel): Promise<void> {
        return $axios.$delete(`${api}/removeAuditory`);
    }
}

```

```
export const auditoryService = new AuditoryService();
```

Б.23 Файл auth-service.ts

```
import { $axios } from '@utils/api';
import { LoginModel } from '~/models/userProfile/LoginModel';
import { RegistrationModel } from '~/models/userProfile/RegistrationModel';
```

```
const api = 'api/auth';
```

```
class AuthorizationService {
  register(data: RegistrationModel): Promise<string> {
    return $axios.$post<string>(`${api}/registration`, data);
  }

  login(data: LoginModel): Promise<string> {
    return $axios.$post<string>(`${api}/login`, data);
  }
}
```

```
export const authorizationService = new AuthorizationService();
```

Б.24 Файл chat-service.ts

```
import { $axios } from '@utils/api';
import { ChatsListModel } from '@models/chat/ChatsListModel';
import { LastMessageModel } from '~/models/chat/LastMessageModel';
```

```
const api = 'api/chat';
```

```
class ChatService {
  createNewSingleChatAndGetNewChatId(userId: number): Promise<number> {
    return $axios.$post<number>(`${api}/createNewSingleChatAndGetNewChatId?userId=${userId}`);
  }

  getAllUsersChatsAsync(): Promise<Array<ChatsListModel>> {
    return $axios.$get<Array<ChatsListModel>>(`${api}/getAllUsersChatsAsync`);
  }
}
```

```

    getAllChatMessages(chatId: number): Promise<Array<LastMessageModel>> {
        return $axios.$get<Array<LastMessageModel>>(`${api}/getAllChatMessages?chatId=${chatId}`);
    }
}

export const chatService = new ChatService();

```

Б.25 Файл group-service.ts

```

import { GroupsMainModel } from '@models/group/GroupsMainModel';
import { GroupMainFullModel } from '~/models/group/GroupMainFullModel';
import { $axios } from '~/utils/api';

const url = 'api/groups';

class GroupService {

    public getAllGroups(): Promise<Array<GroupsMainModel>> {
        return $axios.$get<Array<GroupsMainModel>>(`${url}/getAllGroups`);
    }

    public addNewGroup(groupName: string): Promise<void> {
        return $axios.$post<void>(`${url}/addNewGroup?groupName=${groupName.toString()}`);
    }

    public getGroupById(groupId: number): Promise<GroupMainFullModel> {
        return $axios.$get<GroupMainFullModel>(`${url}/getGroupById?groupId=${groupId}`);
    }
}

export const groupService = new GroupService();

```

Б.26 Файл image-service.ts

```

import { ECloudinaryFolders } from '~/core/enums/ECloudinaryFolders';
import { $axios } from '~/utils/api';

const api = '/api/image';

```

```
class ImageService {

    UploadNewFile( file: FormData, type: ECloudinaryFolders ): Promise<string> {
        return $axios.$post<string>(`${api}/UploadUserProfilePhoto?type=${type}`, file);
    }
}

export const imageService = new ImageService();
```

Б.26 Файл schedule-service.ts

```
import { CreateSchedule } from '@models/schedule/createSchedule';
import { $axios } from '@utils/api';
import { GetScheduleModel } from '@models/schedule/getScheduleModel';
import { ScheduleFilterModel } from '@models/schedule/scheduleFilterModel';

const api = 'api/schedule';

class ScheduleService {

    public createSchedule(dto: CreateSchedule): Promise<void> {
        return $axios.$post(`${api}/createSchedule`, dto);
    }

    public getScheduleByGroupId(filter: ScheduleFilterModel): Promise<GetScheduleModel> {
        return $axios.$post<GetScheduleModel>(`${api}/getScheduleByGroupId`, filter);
    }
}

export const scheduleService = new ScheduleService();
```

Б.27 Файл subject-service.ts

```
import { $axios } from '@utils/api';
import { AddSubjectModel } from '@models/subjects/AddSubjectModel';
import { ShortSubjectModel } from '@models/subjects/ShortSubjectModel';
import { SubjectFullModel } from '@models/subjects/SubjectFullModel';
import { SubjectFullOneDescModel } from '@models/subjects/SubjectFullOneDescModel';

const api = '/api/subject';
```

```

class SubjectService {
  public addNewSubject(dto: AddSubjectModel): Promise<void> {
    return $axios.$post<void>(`${api}/addNewSubject`, dto);
  }

  public getAllSubjects(): Promise<Array<ShortSubjectModel>> {
    return $axios.$get<Array<ShortSubjectModel>>(`${api}/getAllSubjects`);
  }

  public getFullSubjectInfoBySubjectId(subjectId: string): Promise<SubjectFullOneDescModel> {
    return
$axios.$get<SubjectFullOneDescModel>(`${api}/getFullSubjectInfoBySubjectId?subjectId=${subjectId}`);
  }

  public getFullSubjects(): Promise<Array<SubjectFullOneDescModel>> {
    return $axios.$get<Array<SubjectFullOneDescModel>>(`${api}/getFullSubjects`);
  }
}

export const subjectService = new SubjectService();

```

Б.28 Файл user-profile-service.ts

```

import { $axios } from '@/utils/api';
import { UserProfileModel } from '@/models/userProfile/UserProfileModel';
import { UserSocialLinksModel } from '@/models/userProfile/UserSocialLinksModel';
import { AddGroupsModel } from '@/models/userProfile/AddGroupsModel';
import { SaveUserAccessesModel } from '@/models/userProfile/SaveUserAccessesModel';
import { BaseFilterUserModel } from '@/models/userProfile/BaseFilterUserModel';
import { AddUsersRolesModel } from '@/models/userProfile/AddUsersRolesModel';

const api = 'api/UserProfile';

class UserProfileService {
  public getAllUsers(dto: BaseFilterUserModel): Promise<Array<UserProfileModel>> {
    return $axios.$post(`${api}/getAllUsers`, dto);
  }

  public getUserProfileByUserId(userId: number): Promise<UserProfileModel> {
    return $axios.$get<UserProfileModel>(`${api}/getUserProfileByUserId?userId=${userId}`)
  }
}

```

```

    }

    public changeUserProfile(dto: UserProfileModel) {
      return $axios.$put(`${api}/changeUserProfile`, dto);
    }

    public changeUserProfileSocialLinks(dto: UserSocialLinksModel): Promise<void> {
      return $axios.$put(`${api}/changeUserProfileSocialLinks`, dto);
    }

    public addGroupsToUser(dto: AddGroupsModel): Promise<void> {
      return $axios.$put<void>(`${api}/addGroupsToUser`, dto);
    }

    public addUsersAccesses(dto: SaveUserAccessesModel): Promise<void> {
      return $axios.$post<void>(`${api}/addUsersAccesses`, dto);
    }

    public addUsersRoles(dto: AddUsersRolesModel): Promise<void> {
      return $axios.$post<void>(`${api}/addUsersRoles`, dto);
    }
  }
}

export const userProfileService = new UserProfileService();

```

Б.29 Файл login-main-block.vue

```

<template>
  <v-container @keyup.enter="loginAsync()" class="background" fluid>
    <v-card rounded :color="$vuetify.theme.dark ? 'rgba(0, 0, 0, 0.5)' : 'rgba(255, 255, 255, .5)'" outlined
      elevation="24" class="pa-7 card-block">
      <v-row justify="center">
        <v-col cols="12">
          <v-row justify="center" no-gutters>
            <span class="text-h5 font-weight-bold">{{ $t("YourEmail") }}</span>
          </v-row>
          <v-row no-gutters>
            <v-text-field
              v-model="login"
              :color="$vuetify.theme.currentTheme.current"
              :background-color="$vuetify.theme.currentTheme.primary"

```

```

        outlined
        filled/>
    </v-row>

    <v-row justify="center" no-gutters>
        <span class="text-h5 font-weight-bold">{{ $t("YourPassword") }}</span>
    </v-row>

    <v-row no-gutters>
        <v-text-field
            v-model="password"
            :color="$vuetify.theme.currentTheme.current"
            :background-color="$vuetify.theme.currentTheme.primary"
            :type="showPassword ? 'text' : 'password'"
            :append-icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'"
            @click:append="showPassword = !showPassword"
            outlined
            filled/>
    </v-row>

    <v-row class="mb-2" no-gutters>
        <v-col cols="12">
            <span>{{ $t("IfYouDoNotHaveAnAccountYetPlease") }}</span>
            <nuxt-link
                class="text-lowercase text-h6 font-weight-black text-decoration-none"
                :to="localePath('register')">
                <span
                    :style="`color: ${$vuetify.theme.currentTheme.current}`"
                    {{ $tc("Register", 1) }}
                </span>
            </nuxt-link>
            <span class="text-lowercase">{{ $tc("It", 1) }}</span>
        </v-col>
    </v-row>

    <v-row no-gutters>
        <v-btn @click="loginAsync()" height="60" block>{{ $tc('Login', 1) }}</v-btn>
    </v-row>
</v-col>
</v-row>
</v-card>
</v-container>
</template>

```

```

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator';
import { authorizationService } from '@services/auth-service';
import { LoginModel } from '~/models/userProfile/LoginModel';
import { HubConnectionBuilder } from '@microsoft/signalr';

@Component
export default class LoginMainBlock extends Vue {

  public showPassword: boolean = false;
  public login: string = "";
  public password: string = "";
  public pingInterval: any;
  public checkConnectionsInterval: any;

  public async loginAsync() {
    await this.$auth.login({ data: new LoginModel({
      email: this.login,
      password: this.password
    }) }).then(async (res: any) => {
      this.$authToken.setToken('Bearer ' + res.data)
      // this.$router.push(this.localePath({ name: 'students-profile-index' }));
    }).catch(err => {
      this.$toast.error(this.$t("IncorrectLoginOrPassword").toString())
    })
  }

}

}
</script>

<style scoped>
.background::before {
  border-radius: 8px;
}

.card-block::before {
  position: absolute;
  content: "";
  display: block;
  width: 100%;
  height: 100%;

```

```

background: #fff;
opacity: 0.35;
top: 0;
left: 0;
z-index: -1;
}
</style>

```

Б.30 Файл register-main-block.vue

```

<template>
  <v-container @keyup.enter="loginAsync()" class="background" fluid>
    <v-card rounded :color="$vuetify.theme.dark ? 'rgba(0, 0, 0, 0.5)' : 'rgba(255, 255, 255, .5)'" outlined
elevation="24" class="pa-7 card-block">
      <v-row justify="center">
        <v-col cols="12">
          <v-row justify="center" no-gutters>
            <span class="text-h5 font-weight-bold">{{ $t("YourEmail") }}</span>
          </v-row>
          <v-row no-gutters>
            <v-text-field
              v-model="login"
              :color="$vuetify.theme.currentTheme.current"
              :background-color="$vuetify.theme.currentTheme.primary"
              outlined
              filled/>
          </v-row>

          <v-row justify="center" no-gutters>
            <span class="text-h5 font-weight-bold">{{ $t("YourPassword") }}</span>
          </v-row>
          <v-row no-gutters>
            <v-text-field
              v-model="password"
              :color="$vuetify.theme.currentTheme.current"
              :background-color="$vuetify.theme.currentTheme.primary"
              :type="showPassword ? 'text' : 'password'"
              :append-icon="showPassword ? 'mdi-eye' : 'mdi-eye-off'"
              @click:append="showPassword = !showPassword"
              outlined
              filled/>
          </v-row>

```

```

<v-row class="mb-2" no-gutters>
  <v-col cols="12">
    <span>{{ $t("IfYouDoNotHaveAnAccountYetPlease") }}</span>
    <nuxt-link
      class="text-lowercase text-h6 font-weight-black text-decoration-none"
      :to="localePath('register')">
      <span
        :style=""`color: ${$vuetify.theme.currentTheme.current}`">
        {{ $t("Register", 1) }}
      </span>
    </nuxt-link>
    <span class="text-lowercase">{{ $t("It", 1) }}</span>
  </v-col>
</v-row>

<v-row no-gutters>
  <v-btn @click="loginAsync()" height="60" block>{{ $t("Login", 1) }}</v-btn>
</v-row>
</v-col>
</v-row>
</v-card>
</v-container>
</template>

<script lang="ts">
import { Component, Vue } from 'vue-property-decorator';
import { authorizationService } from '@services/auth-service';
import { LoginModel } from '~/models/userProfile/LoginModel';
import { HubConnectionBuilder } from '@microsoft/signalr';

@Component
export default class LoginMainBlock extends Vue {

  public showPassword: boolean = false;
  public login: string = "";
  public password: string = "";
  public pingInterval: any;
  public checkConnectionsInterval: any;

  public async loginAsync() {
    await this.$auth.login({ data: new LoginModel({

```

```

        email: this.login,
        password: this.password
    }) }).then(async (res: any) => {
        this.$authToken.setToken('Bearer ' + res.data)
        // this.$router.push(this.localePath({ name: 'students-profile-index' }));
    }).catch(err => {
        this.$toast.error(this.$t("IncorrectLoginOrPassword").toString())
    })
}

}
</script>

<style scoped>
.background::before {
    border-radius: 8px;
}

.card-block::before {
    position: absolute;
    content: "";
    display: block;
    width: 100%;
    height: 100%;
    background: #fff;
    opacity: 0.35;
    top: 0;
    left: 0;
    z-index: -1;
}
</style>

```

Б.31 Файл schedule-table-show.vue

```

<template>
    <v-card>
        <v-container fluid>
            <v-row class="mb-3" no-gutters justify="center">
                <v-col cols="auto">
                    <span class="text-h5">{{ header }}</span>
                </v-col>
            </v-row>
        </v-container>
    </v-card>
</template>

```

```

</v-row>
<v-row no-gutters>
  <v-col cols="12">
    <v-data-table hide-default-footer :items="content" :headers="headers">
      <template v-slot:[`item.subject`]="{ item }">
        <span>{{ item.subjectName }}</span>
      </template>
      <template v-slot:[`item.teacher`]="{ item }">
        <span>{{ item.teacherName }}</span>
      </template>
      <template v-slot:[`item.auditory`]="{ item }">
        <span>{{ item.auditoryName }}</span>
      </template>
    </v-data-table>
  </v-col>
</v-row>
</v-container>
</v-card>
</template>

<script lang="ts">
import { Component, Vue, Prop } from 'nuxt-property-decorator'
import { Headers } from '@models/headers';
import { SubjectFullOneDescModel } from '@models/subjects/SubjectFullOneDescModel';
import { EScheduleTypeEnum } from '@core/enums/EScheduleTypeEnum';
import { ShowScheduleModel } from '~/models/schedule/ShowScheduleModel';

@Component
export default class ScheduleTableShowComponent extends Vue {
  @Prop({ type: Array, required: true }) model: Array<ShowScheduleModel>;
  @Prop({ type: String, required: true }) header: string;

  public subjects: Array<SubjectFullOneDescModel> = new Array<SubjectFullOneDescModel>();
  public eScheduleTypeEnum: typeof EScheduleTypeEnum = EScheduleTypeEnum;
  public scheduleType: Array<any> = [
    {
      id: 0,
      type: this.$t(EScheduleTypeEnum[EScheduleTypeEnum.Chisltnik])
    },
    {
      id: 1,
      type: this.$t(EScheduleTypeEnum[EScheduleTypeEnum.Znamenik])
    }
  ]
}

```

```

    },
    {
        id: 2,
        type: EScheduleTypeEnum[EScheduleTypeEnum.All]
    }
];

public headers: Array<Headers> = [
    {
        text: this.$t("Pare").toString(),
        value: "pareCount",
        align: 'start',
        sortable: false,
        width: '10%'
    },
    {
        text: this.$t("Subject").toString(),
        value: "subject",
        align: 'center',
        sortable: false,
        width: '40%'
    },
    {
        text: this.$t("Teacher").toString(),
        value: "teacher",
        align: 'center',
        sortable: false,
        width: '40%'
    },
    {
        text: this.$t("Auditory").toString(),
        value: "auditory",
        align: 'end',
        sortable: false,
        width: '10%'
    },
];

public content: Array<ShowScheduleModel> = new Array<ShowScheduleModel>();

private async mounted() {
    this.model = this.model.sort(this.compareFn)
    let a = [ ...Array(this.model[this.model.length - 1].pareCount).keys() ].map( i => i+1);

```

```

a.forEach(el => {
  this.content.push(new ShowScheduleModel({ pareCount : el}))
});

this.model.forEach(el => {
  this.content.splice(el.pareCount - 1, 1, el)
})
}

public compareFn(a: ShowScheduleModel, b: ShowScheduleModel): number {
  if (a.pareCount < b.pareCount) {
    return -1;
  } else if (a.pareCount > b.pareCount) {
    return 1;
  }
  // a must be equal to b
  return 0;
}
}
</script>

<style scoped></style>

```

ДОДАТОК В
Слайди презентації

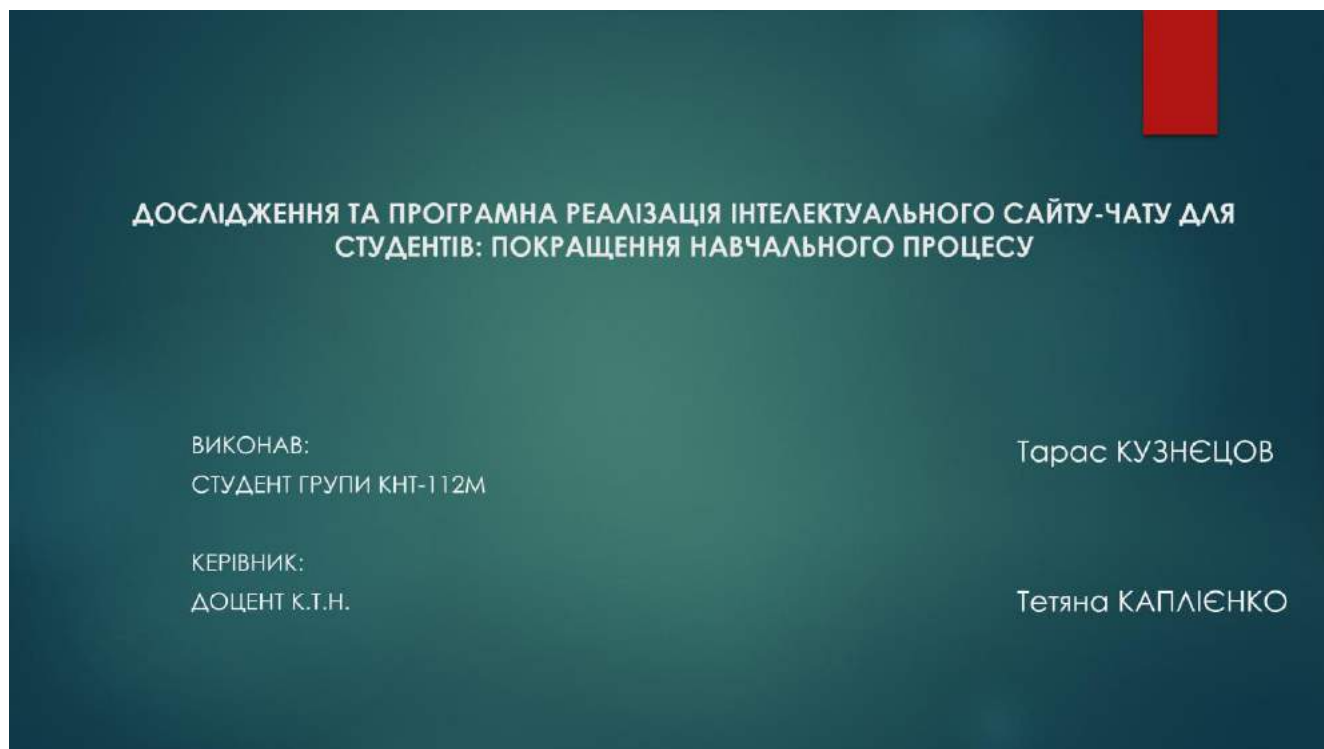


Рисунок В.1 – Слайд № 1

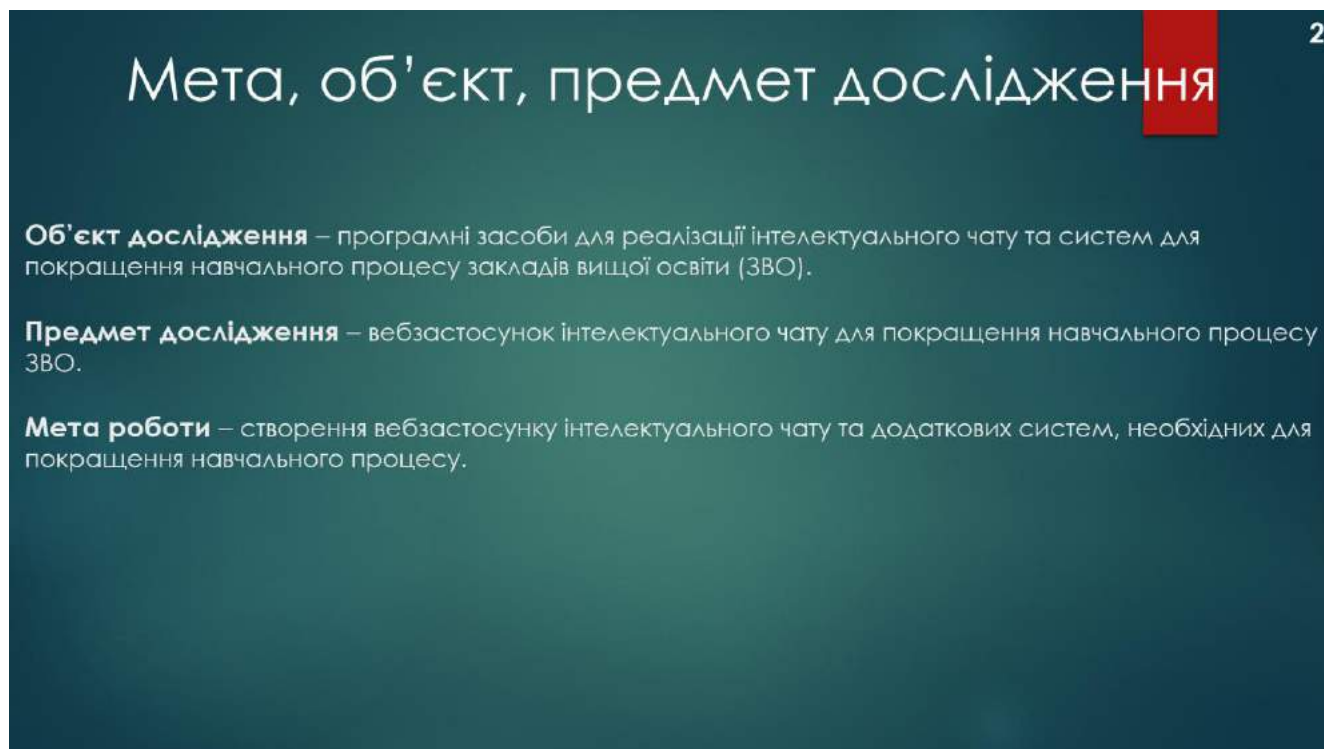


Рисунок В.2 – Слайд № 2

Вибір мови програмування

3

Критерії порівняння	JavaScript	TypeScript	PHP	Python	C#
Мова створювалася спеціально для аналізу даних	+	-	-	+	+
Наявність динамічної типізації	+	++	+	+	+
Можливість статичної типізації	-	+	-	-	++
Наявність зрозумілої та повної документації	+	+	+	+	++
Можливість розробки серверної частини	+	+	+	+	++
Можливість розробки клієнтської частини	+	+	-	-	+
Продуктивність роботи	+	+	+	+-	++
Можливість швидкого підключення сторонніх бібліотек	++	+-	+	+	++
Наявність великої кількості наборів сторонніх пакетів	++	-	+	-	++
Назва пакетного менеджера	NPM	NPM	Composer	Pip	Nuget
Взаємодія з іншими мовами	+	+	+	+	+

Рисунок В.3 – Слайд № 3

Вибір середовища проектування клієнтської частини

4

Фреймворки	React	Vue.js	Angular	Nuxt.js
Низький поріг входження	-	+	-	+-
Підтримка TypeScript	+	++	+	++
Підтримка SPA	+	+	+	+
Наявність зрозумілої та повної документації	+	+	+	++
Легке підключення графічних фреймворків	-	+	-	++
Легке підключення сторонніх бібліотек	+	+	-	++
Продуктивність роботи	+	++	+	++
Управління станами	+	+	+	+
Підтримка SSR	-	-	+	+

Рисунок В.4 – Слайд № 4

Архітектура клієнтської частини вебзастосунку

5

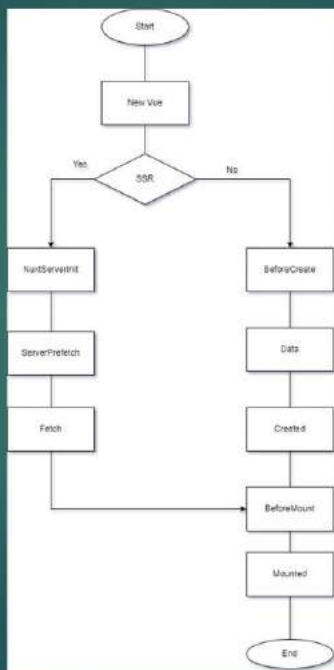


Рисунок В.5 – Слайд № 5

Принцип роботи REST API застосунку

6

REST API Model

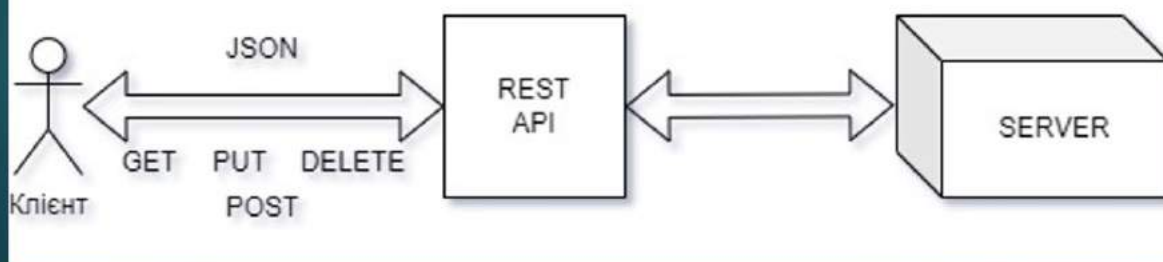


Рисунок В.6 – Слайд № 6

Взаємодія компонентів системи

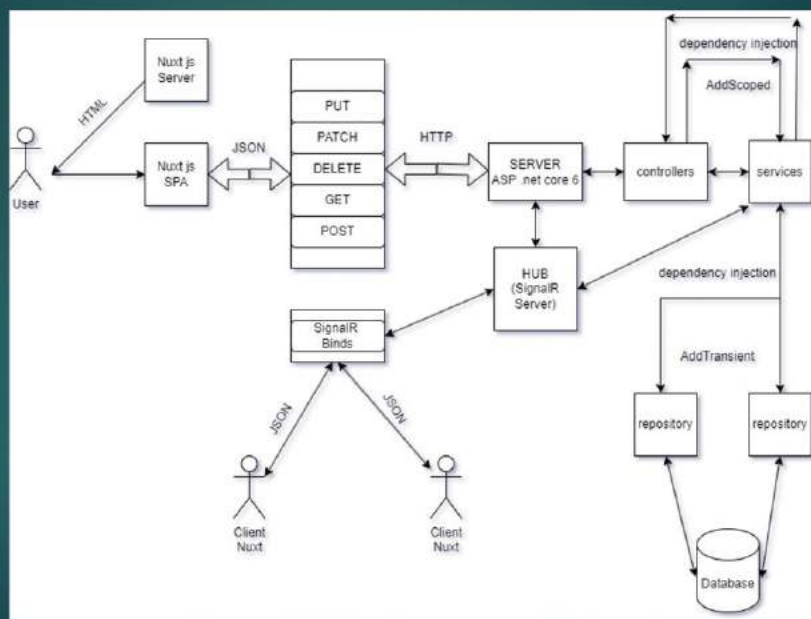


Рисунок В.7 – Слайд № 7

Проектування бази даних

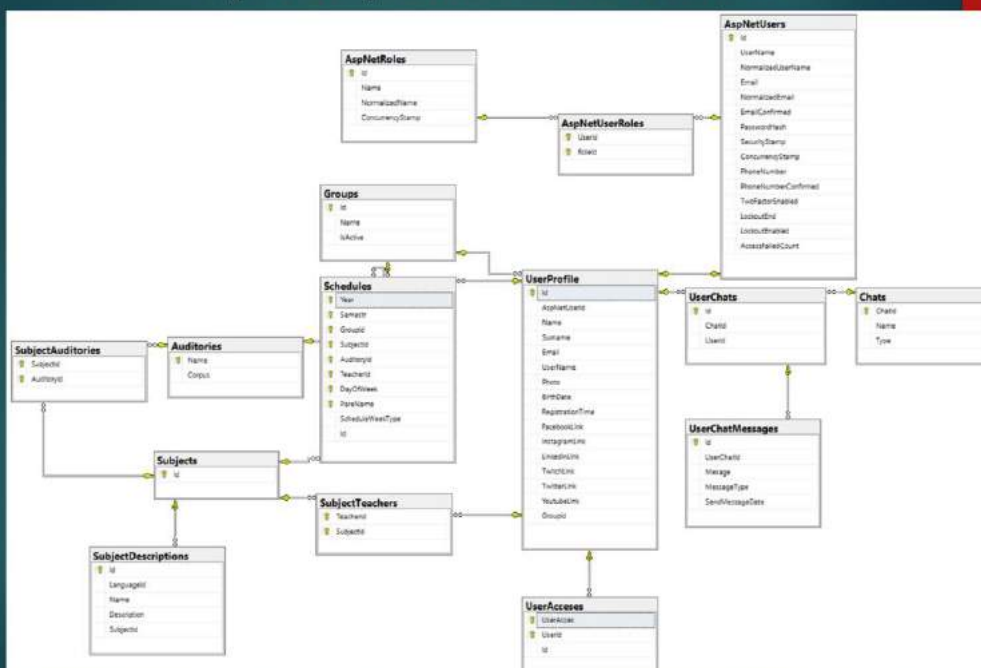


Рисунок В.8 – Слайд № 8

Приклад системних повідомлень оператору

11

Повідомлення українською	Повідомлення англійською	Опис
Неправильний логін чи пароль	Incorrect login or password	Оператор ввів неправильний логін чи пароль під час авторизації
Користувач із такою електронною поштою вже існує в системі	User with the same email are already exist in the system	Оператор ввів електронну пошту, яка вже зареєстрована в системі
Користувач з таким ж ідентифікатором вже існує	User with same userName is already exist	Оператор ввів електронну адресу, яка вже зареєстрована в системі
Така група вже існує	Group is already exist	Оператор ввів назву групи, яка вже є у системі
Ім'я групи не може бути порожнім	Group name can not be empty	Оператор ввів пусту назву групи
Назва групи не може складатися менше ніж з трьох літер	Group name can not be less then three letter	Оператор ввів назву групи, яка складається менш ніж з трьох символів
Необхідно ввести пароль	Password Required	Оператор не заповнив поле пароллю
Електронна пошта обов'язкова	E-mail is required	Оператор не заповнив поле електронної пошти
Обов'язковий пункт	Item is required	Оператор не заповнив один з обов'язкових пунктів

Рисунок В.11 – Слайд № 11

Інтерфейс вебзастосунку

12

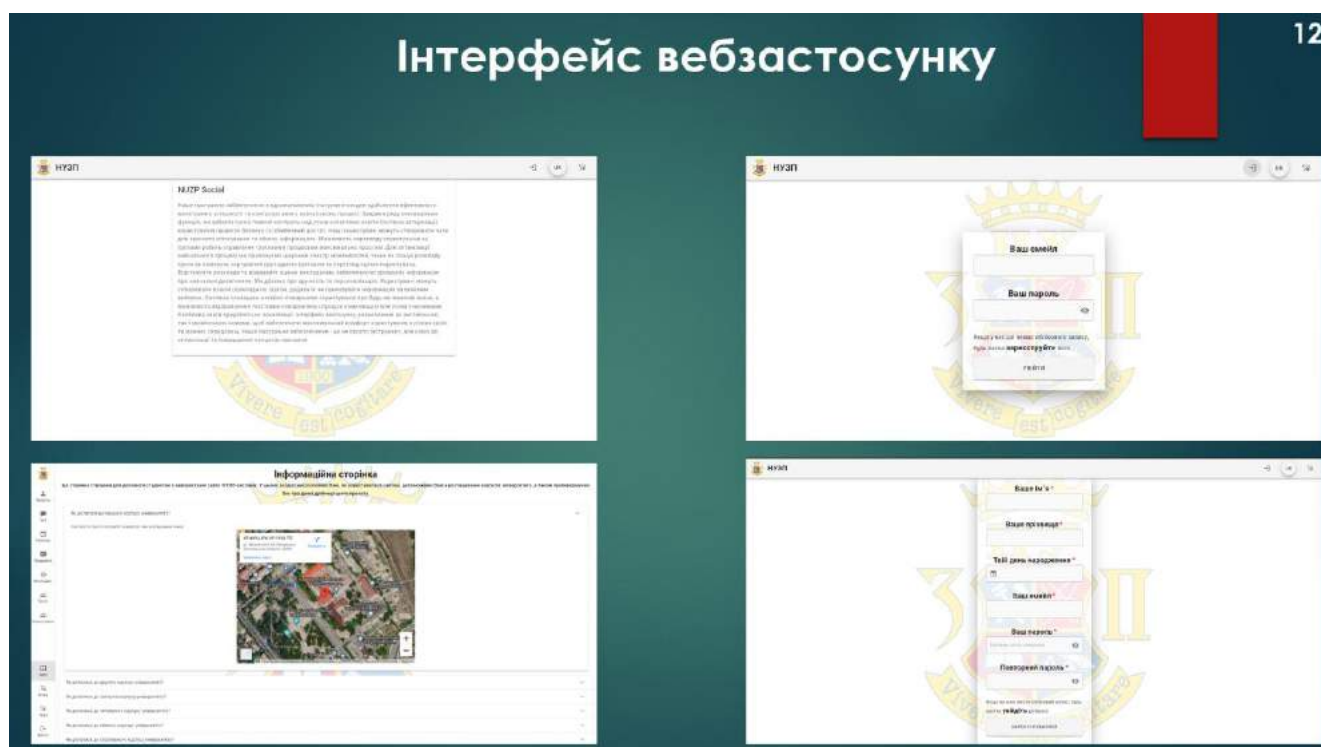


Рисунок В.12 – Слайд № 12

Інтерфейс вебзастосунку

13

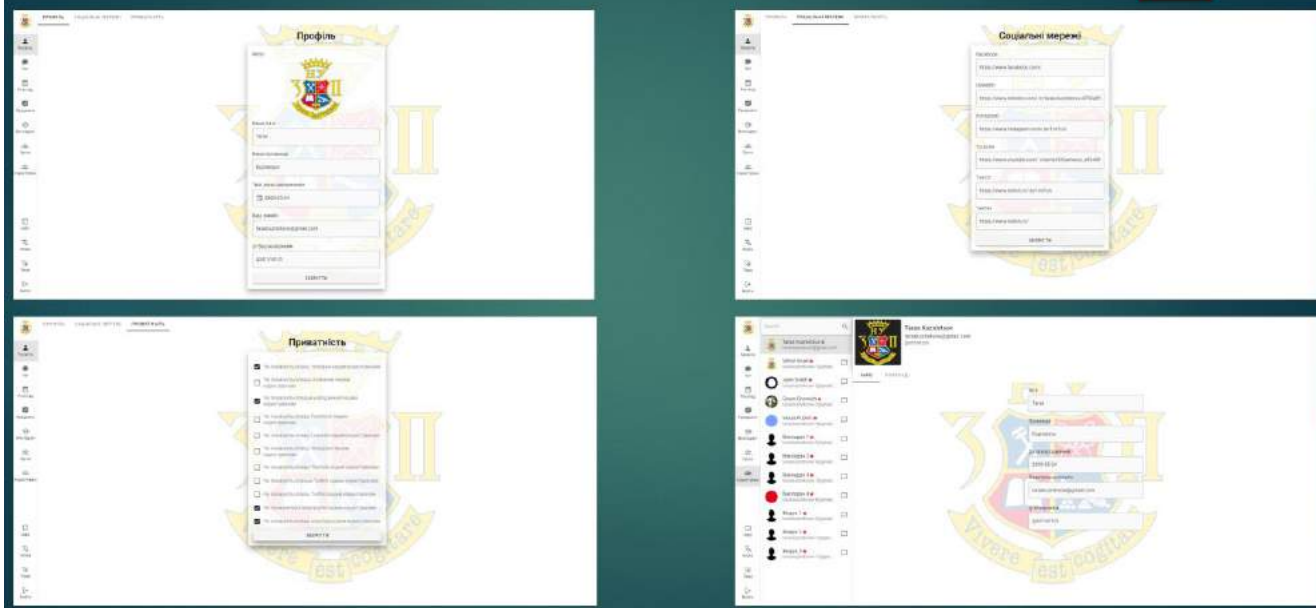


Рисунок В.13 – Слайд № 13

Інтерфейс вебзастосунку

14

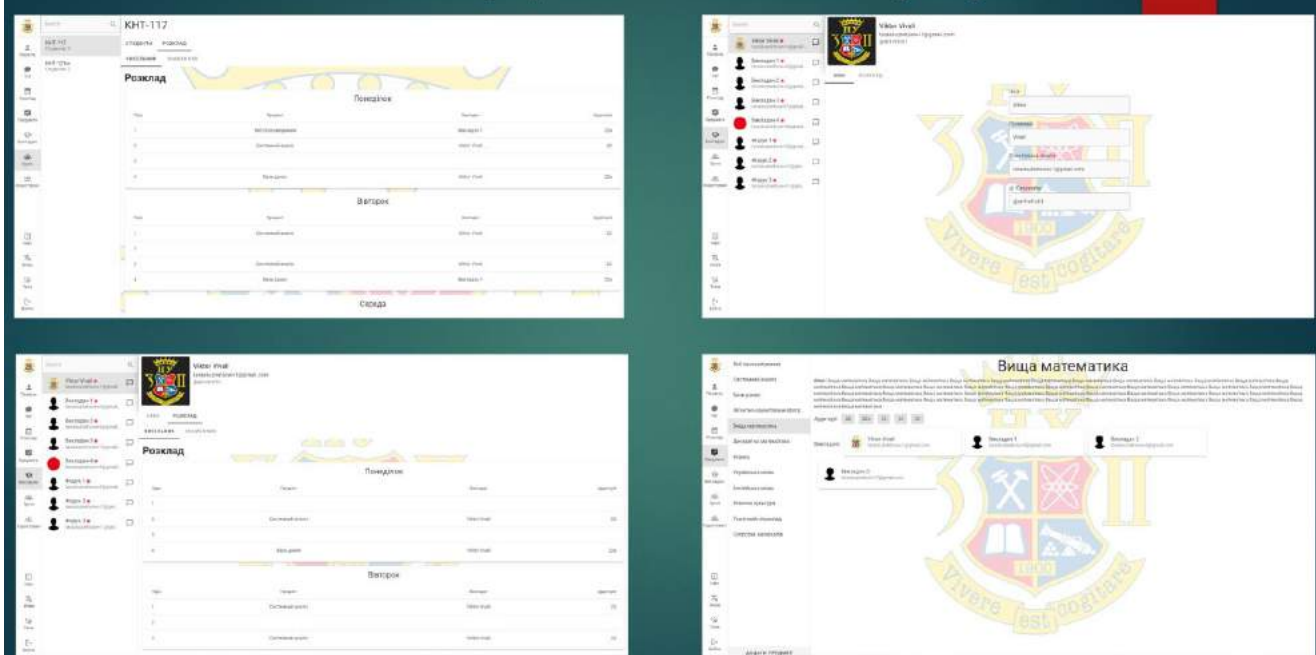


Рисунок В.14 – Слайд № 14

Тестування

15

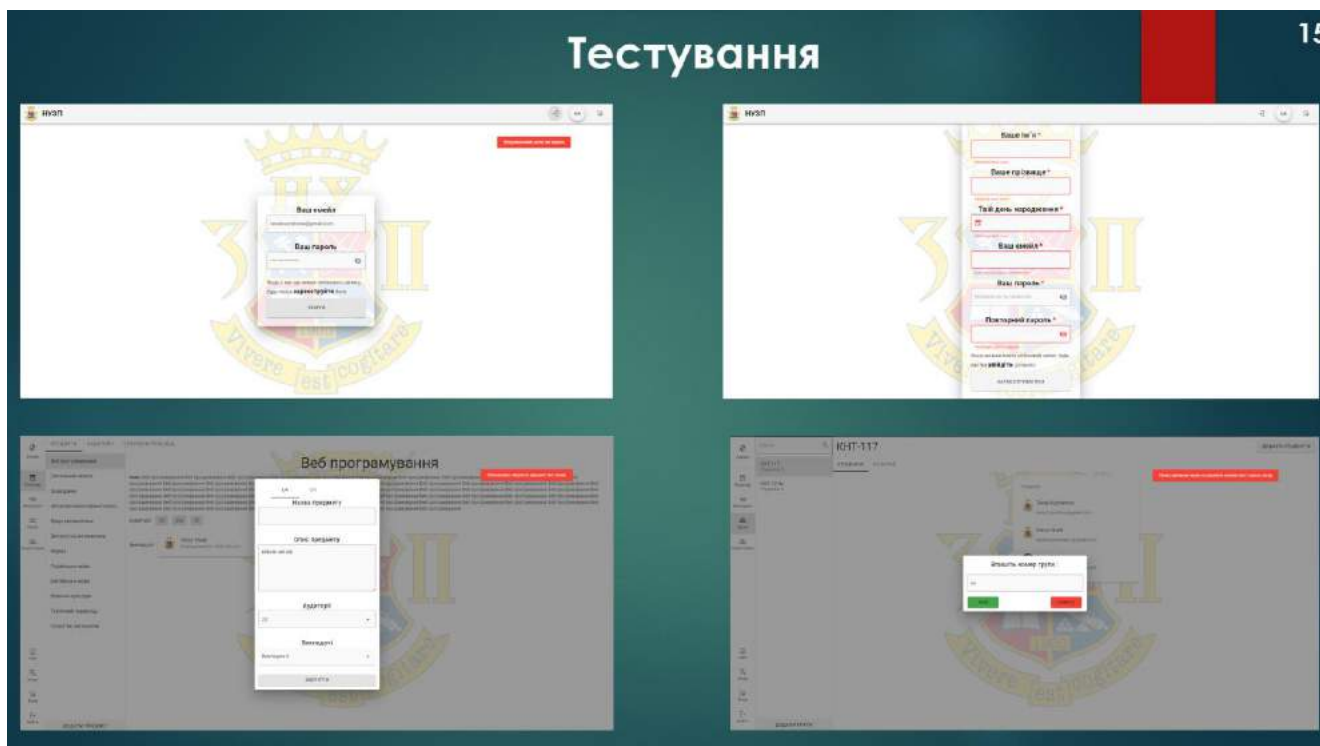


Рисунок В.15 – Слайд № 15

Висновок

16

- Під час виконання роботи було досліджено предметну область, розглянуто проблематику поставленого завдання, наведено його актуальність, доцільність. В даній роботі вивчалася проблема покращення навчального процесу закладів вищої освіти.
- На першому етапі роботи було ретельно проаналізовано предметну область: сформовано проблему та актуальність роботи, розглянуто існуючі методи вирішення розв'язуваної проблеми та готові рішення, виявлено їх переваги та недоліки. Наступним кроком стала розробка архітектури системи, в ході якої було описано структуру розроблюваної програми, обрано інструменти для розробки ПЗ, а також спроектовано майбутню БД. На наступному етапі було реалізовано компоненти програми на основі «Технічного завдання» та надано їхнє обґрунтування.
- Було проведено порівняльний аналіз основних методів для вирішення цього завдання та встановлено, що існуючі засоби не відповідають повному функціоналу, якій потрібен для вирішення завдання дипломної роботи. Тому виникає необхідність в реалізації нового вебзастосунку з розширеним функціоналом.
- Розроблено вебзастосунок для зручної комунікації між учасниками навчального процесу (викладачами та студентами). Серверна частина створеного ПП було написаний на мові програмування C#. Клієнтська частина створеного ПП була написана на мові програмування JS. У якості середовища розробки клієнтської частини було використано Visual studio code, а для серверної Visual studio. Для роботи з базою даних була використана бібліотека Entity Framework.
- В результаті роботи було отримано програмний продукт, що дозволяє виконати поставлене завдання, а саме – покращення навчального процесу. Роботу програми було протестоване на тестових даних. За необхідності програма може бути модифікована та використана під час розробки більш складної, комплексної системи, що вирішуватиме завдання покращення навчального процесу у ЗВО НУ «Запорізька політехніка».

Рисунок В.16 – Слайд № 16